Michigan Law Review

Volume 88 | Issue 4

1990

# Idea, Process, or Protected Expression?: Determining the Scope of Copyright Protection of the Structure of Computer Programs

Steven R. Englund
*University of Michigan Law School*

# NOTE

## Idea, Process, or Protected Expression?: Determining the Scope of Copyright Protection of the Structure of Computer Programs

Computer programs are literary works entitled to copyright protection under federal law.[1] However, they also embody *ideas* and *processes,*[2] neither of which may be protected by copyright.[3] The bars to copyright protection of ideas and processes are codified in the Copyright Act of 1976 ("1976 Act").[4] They derive from separate lines of cases — the former from cases involving the alleged copying of the plots and themes of various literary works[5] and the latter from cases defining the boundary between copyright law and patent law.[6]

Courts considering the alleged copying of the structure, rather than literal copying of the text, of a computer program have usually concerned themselves with whether protected *expression* or an unprotected *idea* was copied. Courts have seldom suggested that it might be an unprotected *process* that was copied.[7] However, this Note concludes that the legislative history of the 1976 Act indicates that that legislation's drafters envisioned a far more prominent role for the process-expression dichotomy than it has played to date. The *process* inquiry is at least as important as the *idea* inquiry in striking the proper balance between promoting progress in the computer art, by granting incentives to create, and impairing progress, by limiting access to utilitarian innovations.[8] This Note develops and describes complementary techniques for distinguishing both unprotected idea *and* unprotected

1. H.R. REP. NO. 1476, 94th Cong., 2d Sess. 54 [hereinafter HOUSE REPORT] ("The term 'literary works' . . . includes . . . computer programs . . . ."), *reprinted in* 1976 U.S. CODE CONG. & ADMIN. NEWS 5659, 5666.

2. A process is a method (in contrast to a machine) for achieving a particular result. *See* Cochrane v. Deener, 94 U.S. 780, 788 (1877) ("A process is a mode of treatment of certain materials to produce a given result. It is an act or a series of acts, performed upon the subject matter to be transformed and reduced to a different state or thing."); *see also* Tilghman v. Proctor, 102 U.S. 707, 728 (1881) ("A machine is a thing. A process is an act, or a mode of acting. The one is visible to the eye, . . . [t]he other is a conception of the mind, seen only by its effects when being executed or performed. Either may be the means of producing a useful result."). This is the definition of process used in patent law. The term's definition in copyright law is not as clear, although use of the patent definition is appropriate. *See infra* note 55.

3. *See infra* notes 42-56 and accompanying text.

4. 17 U.S.C. § 102(b) (1988).

5. *See infra* notes 43-46 and accompanying text.

6. *See infra* notes 52-55 and accompanying text.

7. *See infra* section II.A.

8. *See infra* sections II.B & II.C.

process from protected expression within the context of computer program structure.[9]

Part I of this Note briefly introduces fundamental aspects of the computer art[10] and relevant concepts of copyright law to provide a working background for the analysis that follows. Courts have sometimes appeared not to be fully aware of important elements of the computer art and the nature of progress in that art; this lack of awareness has sometimes operated to the detriment of the goals of copyright law. Part II discusses the threshold question whether copyright protection of a computer program should ever extend beyond the literal instructions of that program to its structure. To determine the proper scope of protection, this Part examines the relevant case law, legislative history, and policy considerations and concludes that established copyright doctrines support some protection for the structures of computer programs. Part III proposes an approach for determining when the copyright in a computer program has been infringed by the copying of its structure and describes the use of this approach within a conventional copyright infringement analysis. This approach differs from existing attempts to address the issue in two ways. First, it gives full effect to both the idea-expression and process-expression dichotomies. Second, it adheres more closely to traditional copyright doctrine than existing approaches. This Note's approach seeks properly to balance the goal of promoting progress in the computer art by providing an incentive to create new programs with the fear of impairing progress in that art by allowing monopolization of ideas or processes.

## I. THE TECHNOLOGICAL AND LEGAL LANDSCAPE

### A. *An Introduction to Computer Technology*

The notions of the "modularity" and "structure" of computer programs are central to the analysis in the rest of this Note. This section

9. These techniques are based upon established copyright law doctrines. A number of commentators have suggested that the problems inherent in including computer programs in the copyright regime could be solved by adopting various models of *sui generis* protection for computer programs. *E.g.*, Stern, *The Bundle of Rights Suited to New Technology*, 47 U. PITT. L. REV. 1229 (1986) (*sui generis* protection for nonliteral aspects of computer programs); Samuelson, *Creating a New Kind of Intellectual Property: Applying the Lessons of the Chip Law in Computer Programs*, 70 MINN. L. REV. 471 (1985). *But see* Raskind, *The Uncertain Case for Special Legislation Protecting Computer Software*, 47 U. PITT. L. REV. 1131 (1986) (arguing no *sui generis* protection needed). Because established doctrine and existing legislation can adequately serve the goals motivating protection of computer programs, there is no need for *sui generis* protection.

10. This Note uses the phrase "computer art," which is common patent law usage, to refer to the body of knowledge about the design and programming of computer systems. In contrast, "computer science" is an academic discipline primarily concerned with the theory and practice of solving problems with computers. *See* E. HOROWITZ & S. SAHNI, FUNDAMENTALS OF DATA STRUCTURES IN PASCAL 1, 4 (1984).

briefly introduces the computer art and, building upon that foundation, defines modularity and structure.

A "computer program" is defined in the 1976 Act as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[11] Although computers are only capable of *directly* executing "computer instructions,"[12] which may be expressed as combinations of the binary digits zero and one,[13] most programs are written in some "programming language"[14]

---

11. 17 U.S.C. § 101 (1988). This is the only term of the computer art defined in the copyright statute. With the exception of the term "module," *see infra* note 26 and accompanying text, this Note uses technical terms as defined in AMERICAN NATIONAL STANDARDS COMMITTEE, AMERICAN NATIONAL DICTIONARY FOR INFORMATION PROCESSING SYSTEMS (1984) [hereinafter ANSC DICTIONARY]. A variety of synonymous terms and different definitions of these same terms are in common use in the technical community. The ANSC Dictionary defines a computer program as a "sequence of instructions suitable for processing by a computer." *Id.* at 82 (emphasis omitted). Although, by its use of the word "sequence," this definition incorporates a notion of ordering that is only implied in the statutory definition, the two definitions have substantially the same meaning.

The word "software" is sometimes used synonymously with "program." *See, e.g.*, Plains Cotton Co-op. Assn. v. Goodpasture Computer Serv., Inc., 807 F.2d 1256 (5th Cir.), *cert denied*, 484 U.S. 821 (1987); Stern, *Another Look at Copyright Protection of Software: Did the 1980 Act Do Anything for Object Code?*, 3 COMPUTER L.J. 1 (1981); Note, *Defining the Scope of Copyright Protection for Computer Software*, 38 STAN. L. REV. 497, 500 (1986) (adopting the definition of software given below but using the terms interchangeably). However, this Note treats the former as a more inclusive term. Software includes "[p]rograms, procedures, rules, and any associated documentation." ANSC DICTIONARY, *supra*, at 367 (emphasis omitted). This distinction is consistent with statutory usage. The 1976 Act uses "program" throughout, and the Report of the National Commission on New Technological Uses of Copyrighted Works ("CONTU") uses "software" only once, in a heading on the first page. *See* NATIONAL COMMN. ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, LIBRARY OF CONGRESS, FINAL REPORT 1 (1979) [hereinafter CONTU REPORT]. Since software may include instruction manuals, the issue discussed in this Note is truly the copyright protection of programs, not software.

12. An instruction is "executed" when a computer performs that instruction. A program is executed when it is "run" on a computer. *See* ANSC DICTIONARY, *supra* note 11, at 146-47. The actual execution of a program is an electronic process that involves no "understanding" by the computer; a computer simply manipulates electric current according to physical laws. Clapes, Lynch & Steinberg, *Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs*, 34 UCLA L. REV. 1493, 1512-13 (1987) [hereinafter *Silicon Epics*]. For a description of this process, see M. MANO, COMPUTER SYSTEM ARCHITECTURE 137-61 (2d ed. 1982). A somewhat less technical explanation is found in J. WEIZENBAUM, COMPUTER POWER AND HUMAN REASON 75-85 (1976). A complementary, but somewhat different, view is that:

> Computational processes are abstract beings that inhabit computers. As they evolve, processes manipulate other abstract things called *data*. The evolution of a process is directed by a pattern of rules called a *program*. People create programs to direct processes. In effect, we conjure the spirits of the computer with our spells.

H. ABELSON & G. SUSSMAN, STRUCTURE AND INTERPRETATION OF COMPUTER PROGRAMS 1 (1985).

13. *See* ANSC DICTIONARY, *supra* note 11, at 80; *see also* J. LEE, THE ANATOMY OF A COMPILER 18 (2d ed. 1974). Computer instructions tell a computer to perform such simple operations as comparing or adding two binary numbers or jumping to a different place in the program. Programs written in a "computer language" consist of computer instructions. ANSC DICTIONARY, *supra* note 11, at 81. For an example of a program written in a computer language, see Comment, *The Incompatibility of Copyright and Computer Software: An Economic Evaluation and a Proposal for a Marketplace Solution*, 66 N.C. L. REV. 977, 1020 (1988), which reprints the first 168 instructions of a relatively short program of approximately 13,000 computer instructions.

that is more convenient to use. Such programs may only be *indirectly* executed by a computer.[15]

To understand how a computer program comes to have structure and why it is important that the processes implemented by programs and the ideas embodied in programs not be protected by copyright law, it is necessary to have some appreciation of computer programming. "Computer programming" refers to "[t]he designing, writing, and testing of programs."[16] Almost all programs are written with some concern for the efficient use of resources such as memory space and computer, programmer, and user time.[17] Since it will seldom, if ever, be possible to write a program minimizing the use of all of these resources, programmers must choose some balance of these criteria appropriate to the nature of the particular program.[18] There are as many ways to approach the task of writing a program to achieve certain efficiency objectives as there are programmers. It is, however, possible to make some generalizations about the techniques that most programmers use.

---

14. Programming languages include all languages in which programs may be written. *See* ANSC DICTIONARY, *supra* note 11, at 308. Examples of programming languages include "high level languages" such as BASIC and FORTRAN, *see id.* at 175, as well as "assembly languages." *See id.* at 21, 81. This Note often simply refers to programming languages as "languages."

15. These programs written in languages that computers are not capable of directly executing may be called "source programs." *See* ANSC DICTIONARY, *supra* note 11, at 369. They must be translated into "object programs," which consist of computer instructions, in order to be executed by a computer. *See* ANSC DICTIONARY, *supra* note 11, at 271. Source programs written in a high-level language may be translated into equivalent object programs by a type of program called a "compiler." *See* ANSC DICTIONARY, *supra* note 11, at 76-77. Those written in an assembly language may be translated into object programs by a program called an "assembler." *See* ANSC DICTIONARY, *supra* note 11, at 20-21. Both compilation and assembly are described well in C. GEAR, COMPUTER ORGANIZATION AND PROGRAMMING 11 (3d ed. 1980). Object programs may be translated into equivalent assembly language programs by a process known as "disassembly." *See generally* D. KNUTH, THE DESIGN OF A TYPICAL COMPUTER AND ITS ASSEMBLY LANGUAGE (1970); E. SOWELL, PROGRAMMING IN ASSEMBLY LANGUAGE 1-3 (1984).

16. ANSC DICTIONARY, *supra* note 11, at 307; *accord* G. SCHNEIDER, S. WEINGART & D. PERLMAN, AN INTRODUCTION TO PROGRAMMING AND PROBLEM SOLVING WITH PASCAL 1 (2d ed. 1982) [hereinafter PROGRAMMING WITH PASCAL] (Programming is "the entire series of steps involved in solving a problem on a computer."); J. HUGHES & J. MICHTOM, A STRUCTURED APPROACH TO PROGRAMMING 3 (1977). For a more philosophical view, see J. WEIZENBAUM, *supra* note 12, at 108 ("Programming is . . . a test of understanding" and "[t]o write a program is to legislate the laws for a world one first has to create in imagination."). A typical commercial program may require one to three years to develop. R. FARLEY, SOFTWARE ENGINEERING CONCEPTS 9 (1985).

17. *See* R. PRESSMAN, SOFTWARE ENGINEERING 148, 284-86 (1982); C. GEAR, *supra* note 15, at 12-14.

18. For example, it is very important that a library program for calculating square roots designed to be used only by other programs be very fast, since this program may be run many thousand times in the course of a day. Minimizing this program's use of computer time may be worth the cost of a few extra days of programmer time or the cost of using a little more computer memory. In designing a word processing program for a personal computer with a large memory, making the program quick and easy for the end user to learn and use may be the most important consideration. Specialists in computer-human interaction have identified many techniques for achieving that goal. *See* Menell, *An Analysis of the Scope of Copyright Protection for Application Programs,* 41 STAN. L. REV. 1045, 1053-55 (1989).

A programmer begins the process of writing a program by precisely defining the program's "function."[19] A program's function is what the program actually does, in contrast to its "implementation," which is the method, including the instructions, that allow a program to accomplish that function.[20] After the function of a program has been precisely defined, a programmer decomposes that function into simpler constituent problems or "subtasks."[21] Subtasks themselves may be decomposed similarly. This process of decomposition requires skill and judgment on the part of the programmer;[22] decisions made at this stage of the programming process can profoundly affect the quality of the resulting program by determining whether a program will be more or less efficient in its use of computer and human resources.[23] The resulting decomposition is determined both by the nature of the problem and by the approach of the programmer. Different programmers faced with the same problem would ordinarily be expected to

19. P. GILBERT, SOFTWARE DESIGN AND DEVELOPMENT 9-11 (1983); PROGRAMMING WITH PASCAL, *supra* note 16, at 2, 6-9.

20. J. STANKOVIC, STRUCTURED SYSTEMS AND THEIR PERFORMANCE IMPROVEMENT THROUGH VERTICAL MIGRATION 7 (1982). The function of a word processing program is to process words according to the rules set forth in its user's manual; the implementation of that program might employ 100,000 assembly language instructions. The function of a program might also be more mundane, such as calculating the square root of a number. Obviously, the program as ultimately written must implement correctly the function it is designed to serve. References to the function of a program in this Note will assume correct implementation, or if the function of the program as implemented differs from the function of the program as specified, it will indicate that divergence.

21. PROGRAMMING WITH PASCAL, *supra* note 16, at 2-3; Wirth, *Program Development by Stepwise Refinement,* 14 COMM. A.C.M. 221, 226-27 (1971); Yohe, *An Overview of Programming Practices,* 6 COMPUTING SURV. 228-29 (1974). For example, a computerized payroll system might be decomposed into the smaller tasks of reading and verifying the correctness of input data, calculating withholding, printing paychecks and reports, and maintaining year-to-date information. PROGRAMMING WITH PASCAL, *supra* note 16, at 2-3. Criteria for the division of a problem into component parts and two sample decompositions of a simple problem are contained in Parnas, *On the Criteria to Be Used in Decomposing Systems into Modules,* 15 COMM. A.C.M. 1053, 1055 (1972). Parnas' piece, a classic in the computer literature, defines "hierarchical structure" and "decomposition" somewhat more formally than this Note. Any differences in the definitions are, however, not important in applying copyright law.

Decomposing the function has several purposes: it minimizes the risk of small errors causing large problems; it makes the detection and correction of programming errors easier; and it permits the easy replacement of one part of the program or use of a part in a different program. H. ABELSON & G. SUSSMAN, *supra* note 12, at 2.

22. *See Silicon Epics, supra* note 12, at 1535-36. *See generally* Parnas, *supra* note 21. While there have been attempts to develop guidelines for good programming, *see, e.g.,* B. KERNIGHAN & P. PLAUGER, ELEMENTS OF PROGRAMMING STYLE 159-61 (2d ed. 1978), these guidelines have made programming a mechanical process to only a slightly greater extent than guides to writing style, *see, e.g.,* W. STRUNK & E.B. WHITE, ELEMENTS OF STYLE (1959), have made writing a mechanical process. *See* Dijkstra, *On the Cruelty of Really Teaching Computer Science,* 32 COMM. A.C.M. 1398, 1400 1989 (describing software engineering as "The Doomed Discipline"). *But see* Menell, *supra* note 18, at 1056 (many design decisions dictated by guidelines of programming practice).

23. *See* McCabe & Butler, *Design Complexity Measurement and Testing,* 32 COMM. A.C.M. 1415, 1415 (1989); *supra* notes 17-18 and accompanying text.

produce somewhat different decompositions.[24]

Once the function of a program has been decomposed into subtasks, a programmer can determine how best to implement the functions of the subtasks and then write instructions in some programming language to do so.[25] This Note refers to the portion of a program implementing a subtask in a particular computer language as a "module." A module is thus nothing more than a relatively short sequence of instructions in some programming language that, when executed, performs some well-defined function that is one of the subtasks defined in the course of decomposing the ultimate function of the program.[26] The functions of the modules in a program together with each module's relationships to other modules constitute the "structure" of the program.[27]

---

24. In many cases, different decompositions may be equally efficient or efficiency objectives may not be well enough defined to choose between two decompositions based upon the degree to which they meet those objectives.

25. PROGRAMMING WITH PASCAL, *supra* note 16, at 3-4. The implementation of relatively simple subtasks may consist entirely of instructions; more complicated subtasks may be implemented by either individual instructions or simpler subtasks. *See infra* note 26.

The programming process continues on to debugging (removing the inevitable errors from the program), testing, documenting, and performing ongoing maintenance. PROGRAMMING WITH PASCAL, *supra* note 16, at 3-5; R. FARLEY, *supra* note 16, at 267-328. Programming is not a linear process, proceeding neatly from one stage to the next. It is often necessary to return to an "earlier" stage of the design process, and frequently, a number of these steps are performed simultaneously. PROGRAMMING WITH PASCAL, *supra* note 16, at 5-6.

26. *See* J. STANKOVIC, *supra* note 20, at 7. A module may incorporate references to (or "call") modules that implement other tasks. This is a "client" relationship: the called module provides a service to its client, the calling module. *See* Parnas, *supra* note 21, at 1057.

This Note uses the term *module* in this context, for want of a better generic term, in spite of its narrower definition in ANSC DICTIONARY, *supra* note 11, at 246 (module defined as "[a] program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading, *e.g.*, the input to, or output from, an assembler, compiler, linkage editor, or executive routine." (emphasis omitted)). This usage is fairly common in practice. *See, e.g.*, J. STANKOVIC, *supra* note 20, at 7; Parnas, *supra* note 21. For the purpose of this Note, program parts variously referred to as functions, modules, procedures, routines, subprograms, and subroutines are all "modules." Discrete groups of consecutive instructions in a computer language may also constitute modules.

27. This definition is equivalent to that used in leading cases addressing structural infringement. *See, e.g.*, Plains Cotton Co-op. Assn. v. Goodpasture Computer Serv., Inc., 807 F.2d 1256, 1260 (5th Cir.) ("organizational structure . . . outlined . . . in the design specifications"), *cert. denied*, 484 U.S. 821 (1987); Whelan Assocs. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1230 (3d Cir. 1986) (an outline of the solution comprising a series of modules), *cert. denied*, 479 U.S. 1031 (1987); SAS Inst., Inc. v. S & H Computer Sys., Inc., 605 F. Supp. 816-25 (M.D. Tenn. 1985) (describing the programming process much as it is described in the text). *See generally infra* section II.A. This notion of structure is described at some length in *Silicon Epics*, *supra* note 12, at 1524-27.

This definition introduces the notion of a hierarchy of modules. It is this hierarchy, and not that implied in this Note's earlier discussion of the programming process, which may be somewhat different, that is important. *See* Parnas, *supra* note 21, at 1057-58. A program's structure may be represented in a form similar to a business organizational chart with client modules positioned superior to those modules they call and the functions of all the modules indicated in the chart's boxes. *See* McCabe & Butler, *supra* note 23, at 1417-18; Stevens, Myers & Constantine, *Structured Design*, 13 IBM SYS. J. 115 (1974); Bauer, *Software Engineering*, in ADVANCED COURSE ON SOFTWARE ENGINEERING 522, 532-34 (F. Bauer ed. Lecture Notes in Economics and Mathematical Systems No. 81, 1973). A module is said to be at a higher level of a program's

The logic and instructions for implementing a subtask may often come from the work of others. The computer industry has long been characterized by the free sharing of programs and program parts among programmers.[28] Not only does the borrowing of implementation details eliminate the need to "reinvent the wheel" with each new program, but borrowing and improving upon more general ideas is an important way in which the computer art progresses.[29] This is important because, in order to ensure that its decisions in copyright cases involving computer programs promote progress in the computer art, a court should have some understanding of the role of copyright law in promoting progress in that art.

The following section describes the legal principles applicable to copyright cases involving computer programs. The section first briefly

---

structure than another if it appears closer to the top of such a diagram. *See* Parnas, *supra* note 21, at 1057. A good description of structure and structure diagrams is found in S. CAMPBELL, MICROCOMPUTER SOFTWARE DESIGN 46-67 (1984).

When comparing the structures of programs, courts have sometimes relied upon flowcharts, which are graphic representations of the definition or solution of a problem in which symbols are used to represent operations, data, equipment, and the flow of control through the program. ANSC DICTIONARY, *supra* note 11, at 158-59; *see e.g.*, CONTU REPORT, *supra* note 11, at 21 n.109 (comparing the representations of programs); *see also* Pearl Sys. v. Competition Elecs., Inc., 8 U.S.P.Q.2d (BNA) 1520, 1522-23 (S.D. Fla. 1988). Because flowcharts convey information that is different from structure (including information about specific operations, data, computer equipment, and the flow of control through the program), structure diagrams are much better tools for comparing the structures of programs than are flowcharts.

Not all program structures have *simple* hierarchical relations between modules. For example, a module may have more than one client or be its own client. Programs written in certain programming languages, such as Lisp, may have flexible structures. H. ABELSON & G. SUSSMAN, *supra* note 12, at xiii. These complications make consideration of the structure of particular programs more challenging. *See infra* note 164. Even if the relationships between the modules in a program are not complicated, the size of the program and the number of modules and client relationships in it may make it difficult to comprehend the structure of the program. The *Whelan* court used the terms *structure, sequence,* and *organization* interchangeably to refer to what this Note calls *structure.* 797 F.2d at 1224 n.1. As a matter of English usage, the terms have different meanings. The *Whelan* court created unnecessary confusion by equating them. Programs consist of a *sequence* of instructions. *See supra* note 11. Instructions are executed one after another until the end of a module is reached or until some instruction to the contrary. This Note uses *sequence* to refer to the order in which individual instructions appear in the text of a program or module. This Note defines *organization* as the order in which modules appear within the text of a program. *Organization* could also reasonably be defined in the way this Note defines *structure.* If two programs differ only in organization, the division of the function of the programs into subtasks and the sequences of instructions within comparable modules will be the same.

28. Markoff, *A Battle to Make Software Free,* N.Y. Times, Jan. 11, 1989, at D1, col. 3; *see* R. STALLMAN, GNU EMACS MANUAL 221 (4th ed. 1986) (author is writing and giving away the software for a complete computer system because "I consider that the golden rule requires that if I like a program I must share it with other people who like it"); *see also* H. ABELSON & G. SUSSMAN, *supra* note 11, at xii (program libraries); PROGRAMMING WITH PASCAL, *supra* note 16, at 292-95 (same). The aggressive assertion of intellectual property rights is threatening to end this tradition. Kolata, *Mathematicians Are Troubled by Claims on Their Recipes,* N.Y. Times, Mar 12, 1989, at E26, col. 1. Many books containing implementations of common functions are available for programmers' use. *See, e.g.,* ASSOCIATION FOR COMPUTING MACHINERY, COLLECTED ALGORITHMS FROM A.C.M. (1980) (a four-volume set, regularly updated).

29. Menell, *supra* note 18, at 1057; Note, *Copyright Infringement of Computer Programs: A Modification of the Substantial Similarity Test,* 68 MINN. L REV. 1264, 1292 (1984).

describes the history of the protection of computer programs under copyright law. The section then introduces the idea-expression and process-expression dichotomies.

### B. *An Introduction to Copyright Law*

After many years of work on the general revision of the copyright law,[30] Congress enacted the 1976 Act,[31] which granted the owners of copyrights in computer programs whatever rights they enjoyed under statutory and common law on the day before the Act became effective.[32] In 1980, Congress amended the computer program provisions of the 1976 Act to reflect the recommendations of the National Commission on New Technological Uses of Copyrighted Works ("CONTU").[33] Under the amended 1976 Act, the owners of copyrights in computer programs receive exclusive rights that are substantially similar to those received by the other owners of copyrights in

30. In 1955, the Copyright Office began a study of copyright revision that led to the introduction of a draft bill in 1964. That bill is reproduced at 3 COPYRIGHT OFFICE, LIBRARY OF CONGRESS, COPYRIGHT LAW REVISION 1-36 (1964). Technological changes and various amendments to the draft bill delayed the enactment of revision legislation until 1976. HOUSE REPORT, *supra* note 1, at 47-50.

31. Pub. L. No. 94-553, 90 Stat. 2541 (codified as amended at 17 U.S.C. §§ 101-810 (1988)). Unless otherwise specified, all references to statutory sections are to sections of the 1976 Act as codified.

32. Copyright Act of 1976, Pub. L. No. 94-553, § 117, 90 Stat. 2541, 2565 (repealed 1980). The Copyright office has allowed the authors of computer programs to register their works since 1964. CONTU REPORT, *supra* note 11, at 15 (citing Copyright Office Circular 31D (Jan. 1965)). While a decision by the Copyright Office to accept a particular application for registration was prima facie evidence of the identity of the copyright claimant and the circumstances of publication and registration under the Copyright Act of 1909, ch. 320, § 55, 35 Stat. 1075, 1086 (repealed 1976), the Copyright Office acknowledged two reasons for doubting that computer programs actually received any protection under the 1909 Act. CONTU REPORT, *supra* note 11, at 15. Programs might not have been within the scope of the term "writings of an author." Copyright Act of 1909, ch. 320, § 4, 35 Stat. 1075, 1076 (repealed 1976). Machine-readable copies of computer programs might not have been "copies" under the 1909 Act. *See* White-Smith Music Pub. Co. v. Apollo Co., 209 U.S. 1, 17 (1908) (A copy is "a written or printed record of [the protected work] in intelligible notation."). The Copyright Office decision to accept applications for the registration of the copyrights in computer programs is described in detail in Cary, *Copyright Registration and Computer Programs,* 11 BULL. OF THE COPYRIGHT SOCY. OF THE U.S.A. 362 (1964). The substance of copyright and copyright registration under the 1976 Act are described in *infra* note 34.

33. In 1974, Congress created CONTU, a commission comprised of copyright experts, to study the copyright problems arising from the development and widespread use of photocopiers and computers. Act of Dec. 31, 1974, Pub. L. No. 93-573, § 201, 88 Stat. 1873; CONTU REPORT, *supra* note 11, at 1, 3-4. On July 31, 1978, CONTU released its final report. In that report, CONTU recommended adoption of a definition of the term computer program and a provision to replace the stopgap protection of § 117 of the original 1976 Act. CONTU REPORT, *supra* note 11, at 12-14. Congress enacted CONTU's proposals, Copyright Act of 1980, Pub. L. No. 96-517, § 10, 94 Stat. 3015, 3028, with only minor modifications. *Compare* CONTU REPORT, *supra* note 11, at 12 *with* 17 U.S.C. § 101 (1988) (definition of computer program adopted without change) *and* 17 U.S.C. § 117 (1988) ("rightful possessor" changed to "owner" and certain typographic changes made in adoption of other provisions). For a more detailed description of the new § 117, see *infra* note 35.

literary works.[34] These include the rights to reproduce the work, prepare derivative works based upon the work, and distribute copies of the work.[35] The term of a copyright is for the life of the author (or the longest-lived of joint authors) and fifty additional years.[36] Anonymous works, pseudonymous works, and works for hire are protected for a term of seventy-five years.[37]

The copyright in a work is infringed when someone violates any of the exclusive rights granted to the owner of the copyright in the work.[38] The owner of a copyright may enforce these exclusive rights

34. Copyright subsists in "works of authorship fixed in any tangible medium of expression," 17 U.S.C. § 102(a) (1988), without any examination or registration. A copyright notice need not be affixed to works first published on or after March 1, 1989, although the use of a notice substantially limits the possibility of the assertion of an "innocent infringer" defense in a subsequent infringement action. 17 U.S.C. § 401 (1988); *see* Berne Convention Implementation Act of 1988, Pub. L. No. 100-568, § 7(a)(4), 102 Stat. 2853, 2857.

One must at least attempt registration of a work originating inside the United States before commencing an infringement action involving that work. 17 U.S.C. § 411 (1988). Registration is also a prerequisite to certain remedies for infringement. 17 U.S.C. § 412 (1988). Registering the copyright in a computer program simply requires submitting to the Copyright Office of the Library of Congress a completed Form TX, 37 C.F.R. § 202.3(b)(2) (1989), which is a single page; the $10 registration fee, 37 C.F.R. § 202.3(c)(2) (1989); and, in most cases, a copy of the first and last 25 pages of either the source or object version of the program. 37 C.F.R. § 202.20(c)(2)(vii)(A)(B) (1989).

35. 17 U.S.C. § 106(1)-(3) (1988). The owners of copyrights in literary and certain other types of works also receive the exclusive right to perform and display their works publicly. 17 U.S.C. § 106(4)-(5) (1988). All owners' exclusive rights are limited by §§ 107-109 (fair use, library reproduction, and first sale doctrine). The subject-matter limitation of § 102(b), *see supra* note 4 and accompanying text, applies to all works, but raises more troubling questions for computer programs than for other literary works. *See infra* Part II.

Section 117 provides that the owner of a copy of a computer program may copy or adapt that program in order to use it or for archival purposes. *See generally* R. NIMMER, THE LAW OF COMPUTER TECHNOLOGY ¶ 1.11[1] (1985). Exact copies must be transferred with the original or destroyed. Adaptations may be transferred only with the consent of the copyright owner. 17 U.S.C. § 117 (1982). This section is of great theoretical importance since a program is copied into a computer's memory whenever it is executed. *See* Vault Corp. v. Quaid Software, Ltd., 847 F.2d 255, 261 (5th Cir. 1988) (loading a program into a computer's memory is copying privileged under § 117). *See generally* Recent Developments, Vault Corp. v. Quaid Software Ltd.: *Limits to Copyright Protection for Computer Programs,* 64 TUL. L. REV. 270 (1989). However, it does not offer a defense to one making or selling large numbers of copies of a program, which is the sort of behavior likely to provoke an infringement action. It is thus largely irrelevant to the discussion in the rest of this Note.

36. 17 U.S.C. § 302(a)-(b) (1988).

37. 17 U.S.C. § 302(c) (1988).

38. 17 U.S.C. § 501(a) (1988). All of these exclusive rights are limited to the work itself. It is not copyright infringement to create a substantially similar work independently. Independent creation of a computer program might be achieved through the use of a "clean room" approach, whereby one team of programmers with access to the original program creates a design specification and a second team without access to or knowledge of the original program (*i.e.,* in a "clean room") creates a new program from the specification. Other than the design specification, there must be no communication about the program between the teams. The inclusion of too much detail in the specification or failure to deprive the second team of access to or knowledge of the first program would be fatal to such an effort. Davidson, *Common Law, Uncommon Software,* 47 U. PITT. L. REV. 1037, 1096 n.151 (1986); Davis, *IBM PC Software and Hardware Compatibility,* COMPUTER LAW., July 1984, at 11, 16-17. For an example of the use of a clean room to create evidence of idea-expression merger, see NEC Corp. v. Intel Corp., 10 U.S.P.Q.2d (BNA) 1177, 1188-89 (N.D. Cal. Feb. 6, 1989).

by bringing an infringement action in federal court.[39] The plaintiff in such an action must show ownership of the copyright in the work in question[40] and that the defendant copied protected aspects of the work.[41]

Section 102(b) of the 1976 Act provides that copyright protection does not "extend to any idea, procedure, process, system, [or] method of operation . . . ."[42] This statutory provision codifies several judicially-developed doctrines. The first is that copyright protects the expression of ideas rather than the ideas themselves.[43] This doctrine is commonly called the "idea-expression dichotomy."[44] The jurisprudence of the idea-expression dichotomy has been refined largely in cases involving novels and plays; its application to such literary works has in many cases allowed the protection of specific plot devices and well-developed characters.[45] The general plot and themes of such

39. 17 U.S.C. § 501(b) (1988).

40. A plaintiff owns the copyright in a work if: the author met the copyright law's low standard of originality; copyright may subsist in the work; the author's citizenship status entitles him to claim a copyright; the statutory formalities have been fulfilled; and the plaintiff is the author of the work or has otherwise been constituted the valid copyright claimant. 3 M. NIMMER, NIMMER ON COPYRIGHT § 13.01[A] (1988). Since a registration certificate is prima facie evidence of the validity of the copyright, 17 U.S.C. § 410(c) (1988), a plaintiff need ordinarily provide only the certificate and evidence of his chain of title from the registrant to establish a prima facie case of ownership. M. NIMMER, *supra*, § 13.01[A].

41. Sid & Marty Krofft Television Prods. v. McDonald's Corp., 562 F.2d 1157, 1162 (9th Cir. 1977); Reyher v. Children's Television Workshop, 533 F.2d 87, 90 (2d Cir.), *cert. denied*, 429 U.S. 980 (1976); Universal Athletic Sales Co. v. Salkeld, 511 F.2d 904, 907 (3d Cir. 1975); Scott v. WKJG, Inc., 376 F.2d 467, 469 (7th Cir. 1967); 3 M. NIMMER, *supra* note 40, § 13.01[B]. Proof of copying is discussed in detail in section III.C.

42. 17 U.S.C. § 102(b) (1988). Section 102(a) provides that "[c]opyright protection subsists . . . in original works of authorship . . . ." 17 U.S.C. § 102(a) (1988). Section 102(b) should thus be read as concerning only the scope of protection of works of authorship in which copyright subsists under § 102(a). The idea-expression and process-expression dichotomies and the merger doctrine limit the range of works that might be found to infringe a work but do not deny that work its copyright. 1 M. NIMMER, *supra* note 40, §§ 2.03[D], 2.18[C][1]; *see also*, 3 *id.* § 13.03[A][1].

Most courts have considered the idea-expression dichotomy in the context of infringement. *See, e.g.*, Atari, Inc. v. North Am. Phillips Consumer Elecs. Corp., 672 F.2d 607, 615 (7th Cir.), *cert. denied*, 459 U.S. 880 (1982); Sid & Marty Krofft Television Prods. v. McDonald's Corp., 562 F.2d 1157, 1163 (9th Cir. 1977). Several courts, however, have misleadingly spoken as if the idea-expression dichotomy or the merger doctrine might work to deny the subsistence of copyright in a work. *See, e.g.*, Morrisey v. Procter & Gamble Co., 379 F.2d 675, 678-79 (1st Cir. 1967) ("[T]he subject matter would be appropriated by permitting the copyrighting of its expression."); Freedman v. Grolier Enters., 179 U.S.P.Q. (BNA) 476, 478 (S.D.N.Y. 1973) ("[T]o permit the copyrighting of the expression would be to grant the copyright owner a monopoly of the idea.").

43. Whelan Assocs. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1234 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987); 3 M. NIMMER, *supra* note 40, § 13.03[A][1].

44. This phrase is usually used to refer collectively to what this Note calls the idea-expression and process-expression dichotomies, two concepts that this Note treats separately. *See, e.g.*, HOUSE REPORT, *supra* note 1, at 57, *reprinted in* 1976 U.S. CODE CONG. & ADMIN. NEWS 5659, 5670 (After the enactment of § 102(b), "the basic dichotomy between expression and idea remains unchanged."). This Note distinguishes the two doctrines because they are in fact different and both are applicable to computer programs for different reasons and in different ways.

45. *See, e.g.*, Twentieth Century-Fox Film Corp. v. MCA, Inc., 715 F.2d 1327, 1329 (9th

works are not protected.[46]

In *Nichols v. Universal Pictures Corp.*,[47] Judge Hand proposed an approach, often called the levels of abstraction test, for distinguishing idea from expression. He wrote:

> Upon any work, and especially upon a play, a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of what the play is all about, and at times might consist only of its title; but there is a point in this series of abstractions where they are no longer protected, since otherwise the playwright could prevent the use of his "ideas," to which, apart from their expressions, his property is never extended.[48]

Unfortunately, as Judge Hand observed, "[n]obody has ever been able to fix that boundary, and nobody ever can."[49] Professor Chaffee made an effort to do so with his "pattern" test. He wrote that "the protection covers the 'pattern' of the work . . . the sequence of events and the development of the interplay of the characters."[50] However imprecise, these are the principal analytical tools for distinguishing ideas from expression.[51]

A second judicially developed doctrine codified in section 102(b) is the *process*-expression dichotomy, which denies copyright protection to the processes embodied in utilitarian works.[52] This doctrine has its

---

Cir. 1983) (plot similarities raise question of fact as to whether *Battlestar: Galactica* copied *Star Wars* expression); Sid & Marty Krofft Television Prods. v. McDonald's Corp., 562 F.2d 1157, 1167 (9th Cir. 1977) (similarity of "total concept and feel" between McDonaldland and H.R. Pufnstuf characters sufficient to find infringement); Sheldon v. Metro-Goldwyn Pictures Corp., 81 F.2d 49, 54-56 (2d Cir.) (similarity of plot and characters of two plays sufficient to find infringement), *cert denied,* 298 U.S. 669 (1936).

46. *See, e.g.,* Burroughs v. Metro-Goldwyn-Mayer, Inc., 683 F.2d 610, 627-28 (2d Cir. 1982) (Tarzan movie did not copy expression of book except as authorized by license); Deller v. Samuel Goldwyn, Inc., 150 F.2d 612, 612 (2d Cir. 1945) (per curiam) (general theme, plot, and ideas may be freely copied), *cert. denied,* 327 U.S. 790 (1946); Shipman v. R.K.O. Radio Pictures, Inc., 100 F.2d 533, 536-38 (2d Cir. 1938) ("sequence of events" protected by copyright, but no substantial similarity thereof in this case); Miller Brewing Co. v. Carling O'Keefe Breweries of Canada, Ltd., 452 F. Supp. 429, 439 (W.D.N.Y. 1978) (copyright protects specific thematic concepts, events, and characterization, not general themes and plot); Midas Productions, Inc. v. Baer, 437 F. Supp. 1388, 1390 (C.D. Cal. 1977) (general themes and plot not protected; no infringement where plot differences outweigh eight specific similarities).

47. 45 F.2d 119 (2d Cir. 1930), *cert. denied,* 282 U.S. 902 (1931).

48. 45 F.2d at 121.

49. 45 F.2d at 121. Judge Hand, author of the opinions in a number of leading copyright cases, wrote, "Obviously, no principle can be stated as to when an imitator has gone beyond the 'idea,' and borrowed its 'expression.' Decisions must therefore inevitably be *ad hoc.*" Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960); *cf.* Knowles & Palmieri, *Dissecting* Krofft: *An Expression of New Ideas in Copyright?,* 8 SAN. FERN. V. L. REV. 109, 126 (1980) (no meaningful distinction between idea and expression).

50. Chaffee, *Reflections on the Law of Copyright,* 45 COLUM. L. REV. 503, 513-14 (1945).

51. *See Silicon Epics, supra* note 12, at 1550-52.

52. 1 M. NIMMER, *supra* note 40, § 2.18[D]; Liebman, Katsch & Leitch, *Back to Basics: A Critique of the Emerging Judicial Analysis of the Outer Limits of Computer Program "Expression,"* COMPUTER LAW., Dec. 1985, at 1,5 [hereinafter *Back to Basics*].

origins in the venerable case of *Baker v. Selden.*[53] In that case, the Court addressed whether an accounting system was protected by the copyright in a book containing an essay describing that system and blank forms for practicing it.[54] The Court distinguished a work describing an "art" (which we would call a method or process) from the art itself and had little difficulty finding the accounting system unprotected.[55]

A third doctrine, closely related to both the idea-expression and process-expression dichotomies, is the merger doctrine. This doctrine denies copyright protection to expression that is inseparable from the ideas or processes underlying the expression.[56]

Cases decided under the 1976 Act have established that copyright protection extends to any type of computer program[57] regardless of the language in which it is written[58] or the medium in which it is em-

---

53. 101 U.S 99 (1880).

54. 101 U.S. at 101 ("[T]he question is, whether the exclusive property in a system of book-keeping can be claimed, under the law of copyright, by means of a book in which that system is explained?").

55. 101 U.S. at 104. The Court's subsidary holding that the blank forms contained in Selden's book were not protected, 101 U.S. 104-07, although better remembered, is probably less important.

This doctrine has been applied to various processes in numerous other cases. *E.g.,* Brief English Systems, Inc. v. Owen, 48 F.2d 555, 556 (2d Cir.) (process for condensing words in system of shorthand not protected by copyright in book describing it), *cert. denied,* 283 U.S. 858 (1931); Burk v. Johnson, 146 F. 209, 213 (8th Cir. 1906) (mutual burial association's business methods not protected by copyright in articles of association and bylaws); Midway Mfg. Co. v. Bandai-America, Inc., 546 F. Supp. 125, 148 (D.N.J. 1982) (rules of game, which are abstract rules and play ideas, not protected by copyright); Seltzer v. Sunbrock, 22 F. Supp. 621, 630 (S.D. Cal. 1938) (system for conducting roller derby not protected by book describing the system).

The term process has not been clearly defined in copyright law, but the patent law definition, *see supra* note 2, is consistent with the term's use in copyright law. The accounting system at issue in *Baker* satisfied the usual patent definition of the term "process," even though it probably would not be patentable. At certain places in its Report, CONTU seemed to equate patent law processes with copyright law processes. CONTU REPORT, *supra* note 11, at 20, 22; *see also infra* text accompanying note 104. At one point the Report cites Parker v. Flook, 437 U.S. 584 (1978), a patent case, for the proposition that "it is important that the distinction between programs and processes be made clear." CONTU REPORT, *supra* note 11, at 18 & n. 96. In other places, CONTU seems to define process more narrowly. *See infra* notes 107-08 and accompanying text.

56. Morrissey v. Procter & Gamble Co., 379 F.2d 675, 678-79 (1st Cir. 1967) (protection does not extend to expression of simple and straightforward sweepstakes rule); *see* Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 742 (9th Cir. 1971) (impossible to distinguish idea and expression of jeweled bee pin). This doctrine received a subtle twist in Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1253 (3d Cir. 1983) (Apple's operating system protected since possible to write other programs with same function), *cert. dismissed,* 464 U.S. 1033 (1984). The doctrine traditionally has been applied to deny protection to expression when an idea is incapable of alternate means of expression, not to find that a particular work is protected.

57. Several early cases concerned the question whether copyright protects operating systems, which are programs that control the execution of other programs. ANSC DICTIONARY, *supra* note 11, at 275. Appellate courts ultimately concluded that both operating systems and "applications programs" intended for end users may be protected by copyright. Apple Computer, Inc. v. Formula Intl., Inc., 725 F.2d 521, 525 (9th Cir. 1984); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1252 (3d Cir. 1983), *cert. dismissed,* 464 U.S. 1033 (1984).

58. This question has arisen in cases involving object programs and programs written in

bodied.[59] In most of the early actions alleging infringement of the copyright in a computer program, the literal text of the program had been copied verbatim.[60] Recently, authors successfully have claimed copyright protection for such nonliteral aspects of programs as their structures.[61] The extent of the protection afforded to program struc-

"microcode," a type of computer language used to instruct a computer to perform such elementary functions as adding two numbers. Courts have found both protected. *Franklin,* 714 F.2d at 1246-49 (object programs protected); NEC Corp. v. Intel Corp., 10 U.S.P.Q. 2d (BNA) 1177, 1178-80 (N.D. Cal. Feb. 6, 1989) (microcode programs protected); GCA Corp. v. Chance, 217 U.S.P.Q. (BNA) 718, 720 (N.D. Cal. 1982) (copyright in source program protects object program as well).

Some commentators argue that the utilitarian nature of these programs precludes copyright protection. Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form,* 1984 DUKE L.J. 663, 741 ("In machine-readable form, the utility of computer programs cannot be separated from their non-utilitarian aspects, and for this reason . . . they ought to be deemed uncopyrightable."). Some commentators would also withhold copyright protection because programs written in these languages are not readily susceptible to direct human perception. *E.g.,* Samuelson, *supra,* at 705 ("If copyright is permitted to extend to machine-readable programs and there is no requirement that the source code be published, one of the traditional norms of copyright law [disclosure] would be subverted."); Stern, *Another Look at Copyright Protection of Software: Did the 1980 Act Do Anything for Object Code?,* 3 COMPUTER L.J. 1, 11 (1981) ("[T]he statutory definition of copy would seem to be conditioned on human intelligibility.").

59. While programs printed on paper or stored magnetically on a disk or tape are clearly protected, protection of programs embodied in read-only memory ("ROM") chips was initially problematic. A ROM is a semiconductor chip that "contains" a program or data that cannot be altered after the chip is produced. Most courts have had little difficulty in extending copyright protection to programs embodied in computer memory chips. *See, e.g., Franklin,* 714 F.2d at 1249; Tandy Corp. v. Personal Micro Computers, Inc., 524 F. Supp. 171, 174-75 (N.D. Cal. 1981). *But see* Data Cash Sys. v. JS&A Group, 480 F. Supp. 1063, 1067-69 (N.D. Ill. 1979), *affd. on other grounds,* 628 F.2d 1038 (7th Cir. 1980).

Those who would deny protection to programs embodied in ROMs argue that ROMs are as much a physical part of the computer to which they are attached as an engine is of an automobile. Since machines are not copies of their blueprints, some argue that a work cannot be "perceived, reproduced, or otherwise communicated" from a device simply because information can be extracted from it like the design of an engine from the engine itself. *See* CONTU REPORT, *supra* note 11, at 32-33 (dissent of Commissioner Hersey); R. NIMMER, THE LAW OF COMPUTER TECHNOLOGY ¶ 1.03[5][b] (1985); Samuelson, *supra* note 58, at 741-49; Stern, *supra* note 58, at 11-12.

60. *See, e.g., Franklin,* 714 F.2d at 1245 (programs copied nearly in their entirety); Williams Elecs., Inc. v. Artic Intl., Inc., 685 F.2d 870, 872 (3d Cir. 1982) (disputed programs "virtually identical"); *see also* Radcliffe, *Recent Developments in Copyright Law Related to Computer Software,* 4 COMPUTER L. REP. 189, 193 (1985) (In most early cases, 85-100% of the text of defendant's program duplicated the plaintiff's program.). When the law was untested, verbatim copying made sense; efforts to make significant changes in a program while copying it may be both expensive in terms of programmer time and counterproductive, because changes tend to render the programs incompatible and present the opportunity to introduce an error into the program.

61. *See infra* section II.A. Structure is not entirely "nonliteral." While it is less literal than the individual instructions, it organizes the text of the program like chapters, sections, and paragraphs organize the text of a book. It is thus readily discernable from the text. *See generally* 3 M. NIMMER, *supra* note 40, § 13.03[A][1].

Authors also successfully have claimed copyright protection for the audiovisual screen displays produced by their programs. Cases involving the copying of a screen display through misappropriation of the program that produces that display can be analyzed as any other case of program copying. More difficult problems are presented when a similar screen display is produced by a wholly different program. The screen display of a video game may be an audiovisual

ture and the desirability of such protection are addressed in the following Part.

## II. Protecting Computer Program Structure Through Copyright Law

Although a number of courts have concluded that, at least under some circumstances, copyright protects the structures of computer programs, there is considerable disagreement among commentators over whether this is a desirable or correct result. This Part considers whether copyright should protect the structure of computer programs. The first section of this Part reviews the case law on the question. All courts considering the question have found that under certain circumstances program structure may constitute protected expression. The second section of this Part examines the legislative history of the copyright statute and concludes that this result is consistent with that history. The final section of this Part considers protection for program structure in light of the purposes of copyright law and concludes that protection of program structure is consistent with those purposes as long as it is limited by the idea-expression and process-expression dichotomies and the merger doctrine.

### A. *Courts Considering the Alleged Copying of the Structures of Computer Programs*

Several courts have been asked to decide whether the copyright in a computer program protects the structure of that program. Although not all of the courts found that the structure of the program immediately before them was protected, all conceded that in at least some instances the structure of computer programs might be within the scope of copyright protection.

In *Whelan Associates v. Jaslow Dental Laboratory,*[62] the Third Circuit held that copyright affords substantial protection to the structures

---

work separate from the computer program that produces the display (a literary work). *E.g.,* Midway Mfg. Co. v. Artic Intl., 704 F.2d 1009, 1012 (7th Cir.), *cert. denied,* 464 U.S. 823 (1983); Williams Elecs., Inc. v. Artic Intl., Inc., 685 F.2d 870, 873-74 (3d Cir. 1982). The same is true of screen displays with more text than a video game display. Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1131-34 (N.D. Cal. 1986). An exclusively textual screen display may be protected as "a 'compilation' of parameter/command terms." Digital Communications Assocs. v. Softklone Distrib. Corp., 659 F. Supp. 449, 463 (N.D. Ga. 1987); *accord* Manufacturers Technologies v. CAMS, Inc., 706 F. Supp. 984 (D. Conn. 1989). Other cases in which even greater protection is claimed are pending. *E.g.,* Lotus Dev. Corp. v. Paperback Software Intl., No. 87-0076-K (D. Mass. filed Jan. 12, 1987) (claiming copying of "look and feel"). *See generally* Lewis, *All's Not Quiet on the Legal Front,* N.Y Times, Mar. 26, 1989, at F8, col. 1.

Determining the appropriate extent of copyright protection of screen displays is beyond the scope of this Note. For a recent treatment of this problem, see Menell, *supra* note 18, at 1088-102. *See also* Samuelson, *Why the Look and Feel of Software User Interfaces Should Not Be Protected by Copyright Law,* 32 Comm. A.C.M. 563 (1989); Note, *Copyright Protection for Computer Screen Displays,* 72 Minn. L. Rev. 1123 (1988).

62. 797 F.2d 1222 (3d Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987).

of computer programs. The plaintiff Whelan and the defendant Jaslow had arranged for Whelan to write a program for managing the business operations of Jaslow's dental laboratory. Whelan was to own the copyright in the program and Jaslow was to use it and act as Whelan's sales representative. Approximately two years after Whelan delivered her program to Jaslow, Jaslow began selling a program with a similar function written in a different language.[63] At trial, both parties' experts agreed that there were structural similarities between the two programs.[64] The trial court found that while there was little or no similarity between the literal instructions of the programs, because they were written in different languages and generally employed different logic within low-level modules, the structures of the programs were substantially similar.[65] The trial court also noted a similarity between the programs' respective file structures and screen displays.[66] The trial court found Whelan's program protected and her copyright infringed.[67]

On appeal, Jaslow argued that the structure of Whelan's program was part of the idea of the program, not its expression.[68] The court did not choose to apply Judge Hand's levels of abstraction approach[69]

63. 797 F.2d at 1226-27.

64. 797 F.2d at 1228. The court's summary of Whelan's expert's testimony suggests that most of the similarity was at the highest levels of the programs' structures. He testified that five modules, apparently called directly from the main program, had virtually identical functions in both programs. Ironically, it is Jaslow's expert's statement that there were "overall structural similarities" that suggests greater similarity between the programs and similarity at lower levels of the structures. If one program is found to infringe upon the copyright in another when the only similarity between them is structural similarity at a high level, the original program is afforded a very high degree of protection.

65. 797 F.2d at 1228-29.

66. Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 609 F. Supp. 1307, 1322 (E.D. Pa. 1985), *affd.*, 797 F.2d 1222 (3d Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987); *see also* 797 F.2d at 1228 (describing plaintiff's expert's testimony). On appeal, the Third Circuit cited the screen displays as evidence of similarity of program structure. 797 F.2d at 1243-45. While similarity of screen displays may be symptomatic of structural similarity, and is probably thus relevant evidence of structural similarity under the low standard of FED. R. EVID. 401, *see* 797 F.2d at 1244, structurally similar programs may have different screen displays and programs with similar screen displays may have different structures. Comparing the structures of two programs is the only way of positively determining whether or not they are structurally similar. Comparing screen displays at best provides weak circumstantial evidence of structural similarity; if the programs' structures have been compared and structural similarity is the only issue, comparing the programs' screen displays adds nothing. Permitting a finder of fact not skilled in the computer art to observe two programs' readily-understandable screen displays may discourage the finder of fact from a detailed examination of the cryptic texts of the programs. It is thus not clear that a fact-finder should be permitted to look at anything but the text of the programs, and possibly structure diagrams prepared therefrom, when only copying of structure is alleged. *See* FED. R. EVID. 403 (exclusion of relevant evidence on the grounds of prejudice or confusion). *Contra* 797 F.2d at 1245 ("Screen outputs are not so enticing that a trier of fact could not evaluate them rationally and with a cool head.").

67. 609 F. Supp. at 1322.

68. 797 F.2d at 1235.

69. For a brief description of Judge Hand's approach see *supra* notes 47-49 and accompanying text.

or any other analytical tool designed for use on a case-by-case basis in striking the proper balance between idea and expression.[70] Instead, the court attempted to formulate a more general rule. Reading *Baker v. Selden*[71] as "focus[ing] on the end sought to be achieved by Selden's book," the court concluded that "the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea."[72] Applying its newly-formulated rule to the facts, the court found that the program's purpose "was to aid in the business operations of a dental laboratory"[73] and that the structure of the program was not necessary to that purpose. Therefore, the court found the structure of Whelan's program to be protected expression.[74]

Perhaps the single virtue of the *Whelan* rule is that it is easy to apply.[75] The widespread application of the rule is likely to have undesirable consequences. Given the court's broad conception of the purpose of a program as what a user might do with it rather than what the program itself actually does, almost any particular structure could be seen as not necessary to that purpose. The court failed adequately to consider the possibility that broad protection might allow Whelan to make it impossible for others to program a computer efficiently to perform the same function or employ the same process. Although the court purported to consider the need to "create the most efficient and productive balance between protection (incentive) and dissemination

---

70. *See* Ladd & Joseph, *Expanding Computer Software Protection by Limiting the Idea,* 2 J.L. & TECH. 5, 9-10 (1987) ("[T]he Third Circuit eschewed case-by-case analysis in favor of a broad rule . . . . The court justified its rule by discussing only one side of the [policy] balance.").

71. 101 U.S. 99 (1880); *see supra* notes 53-55 and accompanying text.

72. 797 F.2d at 1236 (emphasis omitted). Such a rule is not unprecedented. *See* Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1253 (3d Cir. 1983) ("If other programs can be written or created which perform the same function . . . then the program is an expression of the idea and hence copyrightable."), *cert. dismissed,* 464 U.S. 1033 (1984); *see also supra* note 56 (describing the *Franklin* approach as a variation on the merger doctrine).

73. 797 F.2d at 1238. Providing little or no guidance on how to identify a program's purpose, the court identified a very broad purpose for Whelan's program:

> We do not mean to imply that the idea or purpose behind *every* utilitarian or functional work will be precisely what it accomplishes, and that structure and organization will therefore always be part of the expression of such works. The idea or purpose behind a utilitarian work may be to accomplish a certain function *in a certain way* . . . . There is no suggestion in the record, however, that the purpose of [Whelan's] program was anything so refined; it was simply to run a dental laboratory in an efficient way.

797 F.2d at 1238 n.34 (emphasis in original). The court uses purpose and function very broadly as synonyms to refer more to the value of the program or what a user might do with it than what the program actually does. Because the *Whelan* rule protects all aspects of a program not necessary to the court's broad conception of "purpose," the court appears willing to protect aspects of programs that are necessary to their "functions," as that term is defined in this Note. *See supra* note 20 and accompanying text. To the extent this is true, the *Whelan* rule provides significantly more protection than the pair of tests developed in Part III of this Note, in spite of the rule's superficial similarity to the test for distinguishing idea from expression developed in section III.A.

74. 797 F.2d at 1238.

75. *See* Ladd & Joseph, *supra* note 70, at 11.

of information, to promote learning, culture and development,"[76] the court's rule may thus actually impair progress in the computer art.

The Fifth Circuit has also addressed the question of the protection of the structure of computer programs. The defendant in *Plains Cotton Cooperative Association v. Goodpasture Computer Service*[77] hired former employees of the plaintiff who, possessing design specifications for the plaintiff's program, created a program with a structure similar to that of the plaintiff's program.[78] The programs were written in different languages and both the trial and appellate courts noted a genuine dispute over whether the defendant could have copied the plaintiff's program directly.[79] Because this case was an appeal from the denial of a preliminary injunction, the Fifth Circuit was obligated to affirm the district court unless it found an abuse of discretion,[80] and was reluctant to formulate a new rule on the basis of a partially developed record.[81]

The court briefly addressed the question whether the structure of the plaintiff's program constituted protected expression. Erroneously relying upon *Synercom Technology, Inc. v. University Computing Co.*,[82] the court declined to find clear error, and effectively found the

76. 797 F.2d at 1235.

77. 807 F.2d 1256 (5th Cir.), *cert. denied,* 484 U.S. 821 (1987).

78. 807 F.2d at 1258-59.

79. 807 F.2d at 1260-61.

80. 807 F.2d at 1259.

81. 807 F.2d at 1262.

82. 462 F. Supp. 1003 (N.D. Tex. 1978). *Synercom* concerned the alleged infringement of the "input formats" for a program performing structural analysis. This program accepted several kinds of input data that would ordinarily be punched on cards and fed into a computer. The input formats were simply specifications of the order in which various kinds of input data were to be punched onto the cards and provided to the program. From the perspective of the programmer, it did not much matter what set of formats was used, but once a set of formats was selected and a program employing those formats written, it was necessary for all users of the program to arrange their input data according to the chosen formats. Judge Higginbotham analogized the formats to the "figure-H" pattern of a manual transmission gear shift. Once the "figure-H" pattern is selected and a transmission embodying it built and incorporated into a car, moving the shift lever to the upper left arm of the "H," and nowhere else, puts the car into first gear. Illustrations and descriptions of the "figure-H" pattern may be protected by copyright, but not the pattern itself. 462 F. Supp. at 1013.

An input format may best be thought of as a language, or perhaps more precisely, a system of grammar. One communicates with a program by providing it with data in its own language, here, the input format. Thought of in this way, *Synercom* was an easy case, for there can be no more protection for input formats than for the English language itself.

The *Whelan* court mistakenly thought that its decision was at odds with *Synercom,* 797 F.2d at 1239-40, because it confused programs with data and program structure with data arrangement. Copyright protects programs, and perhaps to some extent, program structure. In contrast, copyright might protect an ordered collection of data as a "compilation," *see* 17 U.S.C. §§ 101, 103 (1988), but it rather clearly does not protect a set of abstract rules for arranging data.

The *Plains Cotton* court mistakenly relied upon *Synercom* stating that input formats were "a level of computer software design more specific than functional design and more general than line-by-line program design . . . ." 807 F.2d at 1262. The design of the input format is separate from and largely irrelevant to the overall design of the program. Like the *Whelan* court, the *Plains Cotton* court failed to distinguish data from programs. The fact that input formats are not

structure of the plaintiff's program unprotected, since the record supported inferences that the programs were for analysis of the cotton market, the programmers were intimately familiar with that market, and many of the similarities between the programs were dictated by the nature of the market.[83] This holding demonstrates that the court was concerned that protecting the structure of the plaintiff's program would prevent others from writing programs for analysis of the cotton market. The court left open the possibility that the structures of programs might be protected by copyright under the proper circumstances.

In *Johnson Controls, Inc. v. Phoenix Control Systems,*[84] the Ninth Circuit recently confronted this same question in the context of an appeal from the grant of a preliminary injunction.[85] The court upheld as not clearly erroneous the district court's determination that the allegedly copied structure of the plaintiff's program likely constituted protected expression.[86] The court thus acknowledged that one may infringe the copyright in a computer program without copying the literal text of the program.[87] Because its review was limited to a determination of whether the district court's finding was clearly erroneous, the court did not discuss the proper means for distinguishing unprotected ideas and processes from protected expression in any detail.

In 1985, the year of the district court decision in *Whelan,* three other district courts decided cases involving the alleged copying of structure and at least a few literal instructions. Although these courts did not discuss structural protection to the same extent as the *Whelan* and *Plains Cotton* courts, all did address the question. Two of these courts found the structure of the plaintiff's program protected and infringed;[88] the third found it unprotected but implied that the struc-

---

protected by copyright says little or nothing about whether the structures of programs are protected by copyright.

83. 807 F.2d at 1262.

84. 886 F.2d 1173 (9th Cir. 1989).

85. 886 F.2d at 1174.

86. 886 F.2d at 1175-76.

87. 886 F.2d at 1175. The court noted that "[a] computer program is made up of several different components, including the source and object code, the structure, sequence and/or organization of the program, and the user interface, and the function, or purpose, of the program." 886 F.2d at 1175 (footnotes omitted). While this statement is plainly at odds with the statutory definition of a computer program, *see supra* note 11 and accompanying text, the court correctly identified the fundamental difficulty that copyright law has with computer programs: they are not simply texts, as the statutory definition suggests, but utilitarian works that have a structure and function and instruct a computer to produce a user interface.

88. SAS Inst., Inc., v. S & H Computer Sys., Inc., 605 F. Supp. 816 (M.D. Tenn 1985), involved copying of both the structure and numerous small portions of the literal text of the plaintiff's program. 605 F. Supp. at 822. The court rejected the defendant's argument that it had copied only ideas and found that the copied structure was protected expression. *See* 605 F. Supp. at 830.

In E.F. Johnson Co. v. Uniden Corp. of Am., 623 F. Supp. 1485 (D. Minn. 1985), a second district court was presented with two programs for controlling radios written in different lan-

tures of some programs might be protected.[89]

In 1988, another district court faced this question and issued a confusing opinion that might promise even greater protection to program structure than *Whelan.* In *Pearl Systems v. Competition Electronics, Inc.,*[90] the plaintiff sought a declaratory judgment that the program controlling the plaintiff's pistol shot timer did not infringe the defendant's copyright in the program controlling its shot timer. Although it is not entirely clear from the court's description of the testimony offered at trial, the programs seem to have been similar only in their use of two modules with similar functions and possibly with similar high-level structures.[91] The court stated that the structures of the two pairs of modules were "nearly identical;"[92] however, it is not

guages that had similar structures and a few literal similarities. The defendant admitted disassembling the plaintiff's object program, transforming it into flowcharts, and using those flowcharts as a guide in developing its own program. Although there seems to have been extensive borrowing of the low-level structure of the plaintiff's program, the approach used by the defendant is not dramatically different from the "clean room" technique for independent creation of a similar program. *See supra* note 38. If flowcharts conveying only unprotected ideas and processes were prepared from the plaintiff's program by people other than those who actually wrote the defendant's program, and there was no other contact between the groups, a finding of infringement would have been improper.

Several errors and idiosyncrasies appeared in both programs. 623 F. Supp. at 1494-96. Courts regard the existence of common errors in similar works as very strong evidence of copying. Cooling Sys. & Flexibles, Inc. v. Stuart Radiator, Inc., 777 F.2d 485, 492 (9th Cir. 1985); Eckes v. Card Prices Update, 736 F.2d 859, 863-64 (2d Cir. 1984).

While line-by-line comparison of the two programs was "meaningless" because they were written in different assembly languages, the court had no difficulty concluding that the programs were substantially similar and that the defendant had copied the plaintiff's program. 623 F. Supp. at 1497.

The court directly confronted the question whether that which was copied was protected by copyright. Because the defendant could have built a radio compatible with the plaintiff's without copying the plaintiff's program to the extent it did, the court found that the defendant had misappropriated the plaintiff's protected expression, including the structure of its program. 623 F. Supp. at 1501-03.

89. In Q-Co Indus., Inc. v. Hoffman, 625 F. Supp. 608 (S.D.N.Y. 1985), the defendants, former employees of the plaintiff, produced a teleprompter program with a function similar to one on which they worked while employed by the plaintiff but running on a different type of computer. 625 F. Supp. at 611-13. Experts testified that there was overall structural similarity between the two programs and that all four of the modules at the highest level of the structure of the defendant's program were structurally similar to four of the twelve modules at the highest level of the structure of the plaintiff's program. There were also a few textual similarities between the programs; however, the two programs were written in different languages and, at least at lower levels of the programs' structures, "employ[ed] wholly distinct algorithms." 625 F. Supp. at 614. The court used reasoning very similar to that of the *Plains Cotton* court, concluding that any teleprompter program would have the same modules and thus, only the idea of plaintiff's program, rather than its expression, was copied. 625 F. Supp. at 616. The court found the structure of the plaintiff's program unprotected. The plaintiff ultimately prevailed, but on a state trade secret claim. 625 F. Supp. at 618.

90. 8 U.S.P.Q.2d (BNA) 1520 (S.D. Fla. 1988).

91. *See* 8 U.S.P.Q.2d (BNA) at 1522-23. The court refers repeatedly to the "system level designs" of the two programs. As the court defines that phrase, it refers to what this Note calls the definition of the function of the program and the initial decomposition of the function into a set of subtasks. *Compare* 8 U.S.P.Q.2d (BNA) at 1522 n.3 *with supra* notes 19-21 and accompanying text.

92. 8 U.S.P.Q.2d (BNA) at 1525.

clear that this court used the term structure as narrowly as have other courts.[93] Although the programs seem to have been similar only in the most general sense, the court found that the plaintiff had copied the defendant's protected expression.[94]

This case expands upon the *Whelan* rule, since in *Whelan,* there was apparently some similarity at all levels of the programs' structures. Here, the court found the copyright in a program infringed by a program that had two modules with functions similar to modules in the first program, may have been structurally similar at the highest level of two modules, and had a similar, simple user interface. The case seems to stand for the proposition that the idea of a program *is* protected by copyright.

All of these courts have concluded that the structure of a program may be protected by copyright under the proper circumstances. All of these cases identify the idea-expression dichotomy as the primary hurdle to finding that copyright protects the structure of computer programs. Most apply some form of the merger doctrine. All of the cases finding protection failed to explicitly consider the possibility that the protection they afford might extend to an efficient process that under traditional doctrines is not protected by copyright. Extending copyright protection to processes was clearly not the intention of either Congress or CONTU, which envisioned a narrower scope of protection than the courts have provided. This Note turns next to the intentions of Congress and CONTU.

### B. *Legislative Materials and Copyright Protection of Computer Program Structure*

The House Report accompanying the 1976 Act and the CONTU Report are the principal texts which evidence the intentions of the authors of the 1976 Act and its 1980 amendments. This section first considers the intention of Congress as expressed in the House Report. The House Report is fairly comprehensive, but only its discussion of section 102 addresses the problem of determining the scope of copyright protection of computer programs, and this discussion is in very

93. *See supra* note 27 and accompanying text. The *Pearl Systems* court apparently understood structure to include function and function's manifestation in the fact that corresponding modules were executed when similarly labeled buttons on the respective shot timers were pushed. *See* 8 U.S.P.Q.2d (BNA) at 1522-25. The court noted that "[a]ny differences in the [modules] . . . are inconsequential, as the Pearl Systems device captures the 'total concept and feel' of the Competition Electronics [modules]." 8 U.S.P.Q.2d (BNA) at 1524 (citations omitted).

94. Purportedly applying the *Whelan* rule to these facts, the court wrote:

> The par time entry subroutine was designed to provide a method for the user to set a par time. That is the idea. The shot review subroutine was designed to allow the user to review the shots he or she has fired and to learn of the time that elapsed between each shot. That is also an idea. The subroutines themselves are expressions of those ideas.

8 U.S.P.Q.2d (BNA) at 1524.

general terms.[95] Because CONTU carefully considered computer program protection and Congress adopted its recommendations almost without change, several courts have considered the CONTU Report a reflection of legislative intent — even with respect to the 1976 Act.[96] Unfortunately, neither the 1976 House Report nor the CONTU Report shed much light on whether those bodies thought program structure should be protected; they do, however, suggest that both bodies favored relatively narrow protection for programs generally. This section concludes that the best interpretation of the legislative history is that neither Congress nor CONTU would find program structure unprotected as a matter of law, but would apply established doctrines on a case-by-case basis to determine whether structure is protected.

The portion of the House Report describing the categories of works of authorship listed in section 102(a) provides that computer programs are included in the literary works category "to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves."[97] This statement is peculiar in its requirement of "expression of original ideas" rather than the "original expression of ideas." Section 102(a) itself only requires "original works of authorship,"[98] not original ideas,[99] and it seems highly unlikely that Congress intended to make wholly different principles applicable to computer programs. Moreover, the idea-expression dichotomy is codified in section 102(b) rather than section 102(a). For these reasons, the only sensible interpretation of this statement is that Congress was adamant that the idea-expression dichotomy be applied to computer programs.

The description of section 102(b)[100] in both the House Report and the much shorter Senate Report directly addresses computer programs:

> Some concern has been expressed lest copyright in computer pro-

95. Only a single paragraph of the House Report accompanying the 1980 Act pertains directly to computer programs, and it merely announces the near-verbatim adoption of CONTU's recommendations. *See* H.R. REP. No. 96-1307, 96th Cong., 2d Sess. 23, *reprinted in* 1980 U.S. CODE CONG. & ADMIN. NEWS 6460, 6482.

96. *See, e.g.*, Apple Computer, Inc. v. Formula Intl., 594 F. Supp. 617, 621 (C.D. Cal. 1984) (CONTU Report most valuable rationale for both 1976 and 1980 Acts); Micro-Sparc, Inc. v. Amtype Corp., 592 F. Supp. 33, 35 n.7 (D. Mass. 1984) (CONTU Report comprises the "entire legislative history" of § 117); Midway Mfg. Co. v. Strohon, 564 F. Supp. 741, 750 n.6 (N.D. Ill. 1983) (same); *cf.* Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1241 (3d Cir. 1986) (CONTU Report "of marginal relevance, at best," when neither § 101 definition of computer program nor § 117 at issue), *cert. denied*, 479 U.S. 1031 (1987).

97. HOUSE REPORT, *supra* note 1, at 54, *reprinted in* 1976 U.S. CODE CONG. & ADMIN. NEWS 5659, 5667.

98. 17 U.S.C. § 102(a) (1988).

99. Nobody could seriously argue that the subsistence of copyright in *West Side Story* should be denied because the underlying idea is at least as old as *Romeo and Juliet*, if not timeless.

100. Section 102(b) provides that copyright protection does not "extend to any idea, procedure, process, system, [or] method of operation . . . ." 17 U.S.C. § 102(b) (1988).

> grams should extend protection to the methodology or processes adopted by the programmer, rather than merely to the "writing" expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law.[101]

This paragraph states nothing more than that the process-expression dichotomy applies to computer programs. The presence of the paragraph in the Report reflects Congress' concern about this issue, but the paragraph is of little help in determining whether program structure in general is an idea, a process, or protected expression. It is of even less help in determining the extent to which the structure of a particular program is merged with an idea or a process. To gain greater insight on these questions, this section turns next to the CONTU Report.

The CONTU Report's discussion of the scope of copyright protection of computer programs is based upon its interpretation of existing doctrines. This discussion is independent of CONTU's recommendations,[102] and so is, in some sense, mere dicta. The Report, however, is deserving of greater attention than other commentary because CONTU based its recommendations, especially its recommendation that Congress repeal old section 117, which provided unique terms for copyright protection of computer programs, upon its understanding of these general principles of copyright law. CONTU's recommendations can only be understood against the backdrop of its discussion of these principles.

Much of CONTU's discussion is devoted to the process-expression dichotomy. CONTU summarizes *Baker* as "stand[ing] for the proposition that using the system does not infringe the copyright in the description," and notes that this is also the rule of section 102(b).[103] Applying this rule to computer programs gives the result that "one is always free to make a machine perform any conceivable process (in the absence of a patent), but one is not free to take another's program."[104] CONTU's emphasis upon *Baker* and its expectation that limiting doctrines[105] would significantly narrow the scope of protec-

---

101. HOUSE REPORT, *supra* note 1, at 57, *reprinted in* 1976 U.S. CODE CONG. & ADMIN. NEWS 5659, 5670; S. REP. No. 473, 94th Cong., 1st Sess. 54 (1976).

102. CONTU REPORT, *supra* note 11, at 18.

103. *Id.* at 19.

104. *Id.* at 20.

105. CONTU specifically refers to the "fruits of intellectual labor" doctrine, *see generally* Trade-Mark Cases, 100 U.S. 82, 94 (1879), which would deny protection to simple and obvious programs, and the merger doctrine. CONTU REPORT, *supra* note 11, at 20; *see supra* note 56 and accompanying text.

tion evidences a genuine concern that both ideas and processes not be protected by copyright.

CONTU devoted significant energy to developing a notion of what a process is. CONTU apparently conceived of processes rather narrowly, relying on the generous application of the merger doctrine to guard against the assertion of exclusive rights in processes. CONTU conceded that it is difficult "to draw the line between the copyrightable element of style and expression in a computer program and the process which underlies it."[106] In addressing this problem, the Report discusses several stages at which a program might be copied: when represented as text on paper, when recorded on a disk, when loaded in a memory, and when being executed.[107] At all of these stages but the last, CONTU finds literal infringement of protected expression a possibility. This discussion concludes that "[t]he movement of electrons through the wires and components of a computer is precisely that process over which copyright has no control."[108]

In determining CONTU's view on the protection of structure, it is unclear how much significance one should attach to its discussion of these opportunities for literal copying. One view is that CONTU did not contemplate that the copying of anything other than literal instructions could ever constitute copyright infringement and therefore regarded structure as a process or possibly an idea as a matter of law. Some commentators derive support for this view from the testimony of Arthur Miller (a member of CONTU's Software Subcommittee) and Arthur Levine (the Executive Director of CONTU) as expert witnesses in *Evergreen Consulting v. NCR Comten, Inc.*,[109] a case of alleged structural infringement that was settled before a decision was rendered.[110] These statements must, of course, be considered in light of the adversarial setting in which they were made. These commentators also find support in CONTU's rejection of proposals that would have explicitly made computer programs derivative works of flowcharts and permitted protection of the ideas expressed in works when those ideas were selected from among a number of alternative ideas.[111] However, the rejection of these proposals is also consistent with the view that structure may constitute protected expression with-

106. CONTU REPORT, *supra* note 11, at 22.

107. *Id.*

108. *Id.*

109. No. 82-5946-KN (C.D. Cal. filed 1982).

110. *Back to Basics, supra* note 52, at 8 (quoting Letter from Arthur R. Miller to Salem M. Katsch, at 7 (Oct. 29, 1985) ("the statutory protection extends only to the statements and instructions that make the machine perform.") and an unspecified declaration by Arthur J. Levine ("CONTU did not want to extend copyright protection for computer programs to things such as algorithms, logic, structure or flow of the program . . . .")).

111. *Id.* at 7.

out the adoption of such dramatic departures from traditional doctrines.

An alternative interpretation is that CONTU never expressed a view on whether program structure should be protected by copyright, but assumed that it might be under the proper circumstances. If so, structure should be protected as part of the expression of the program except to the extent that protection of structure would impair access to ideas or processes. This is necessarily a case-by-case determination.

Support for this alternate view is found in the CONTU majority's express rejection of Vice-Chairman Nimmer's suggestion that programs should be protected only when their use leads to copyrightable output.[112] The majority noted that "[a]lthough the distinction tries to achieve the separation of idea from form of expression, that objective is better realized through the courts exercising their judgement in particular cases."[113] Although this expresses no view on the protection of structure per se, it does indicate CONTU's preference for resolving idea-expression and process-expression questions through the application of established limiting doctrines on a case-by-case basis.

In addition, Vice-Chairman Nimmer's testimony as an expert witness opposing the testimony of Levine and Miller in *Evergreen Consulting* supports the position that CONTU assumed that structure may be protected under the proper circumstances. In *Evergreen Consulting,* Nimmer testified that he believed CONTU understood that the repeal of old section 117 would give the authors of computer programs the same rights as the authors of other literary works.[114] Nimmer understood CONTU to have supported the application to computer programs of doctrines applicable to plays and other literary works.[115] Nimmer argued that:

> CONTU had no views, and made no recommendations which would negate the availability of copyright protection for the detailed design, structure and flow of a program under the copyright principles that make copyright protection available, in appropriate circumstances, for the structure and flow of a novel, a play or a motion picture.[116]

---

112. CONTU REPORT, *supra* note 11, at 27 (concurring opinion of Commissioner Nimmer).

113. CONTU REPORT, *supra* note 11, at 21.

114. Declaration of Melville B. Nimmer, ¶ 7, Evergreen Consulting, Inc. v. NCR Comten, Inc., No. 82-5946-KN (C.D. Cal. filed 1982), *reprinted in Silicon Epics, supra* note 12, at 1586. Vice-Chairman Nimmer reasoned that since two commissioners dissented when CONTU failed to recommend adoption of language in § 117 that would deny protection to computer programs on the same terms as other works, CONTU must have granted protection on the same terms as other works. *Id.* ¶ 8. He also cited numerous scattered passages of the Report in support of his position. The Miller letter, *see supra* note 110, is based upon Commissioner Miller's declaration in opposition to Professor Nimmer in the same action. Vice-Chairman Nimmer responded to a number of Commissioner Miller's arguments. *E.g.,* Declaration of Melville B. Nimmer, *supra,* ¶¶ 20-21, 26.

115. Declaration of Melville B. Nimmer, *supra* note 114, ¶¶ 12, 17, 25, *reprinted in Silicon Epics, supra* note 12, at 1587-89, 1591.

116. *Id.* ¶ 28.

There is not universal agreement that CONTU's thoughts on the problem of structure are deserving of more attention than those of any other commentators,[117] and it is not entirely clear what weight one should attach to the expert testimony of Commissioner Miller, Executive Director Levine, and Vice-Chairman Nimmer in a particular case. However, because of the prominence of the members of CONTU, the depth of its study, and the role of its Report in shaping the development of this area of the law, this Note suggests that one ought to attempt to determine and give serious consideration to CONTU's views. The CONTU Report itself is helpful in determining whether CONTU believed structure should be protected, but doesn't definitively answer the question. Commissioner Miller argued that CONTU believed that structure should never be protected because it is an idea or process, while Vice-Chairman Nimmer declared that CONTU believed that structure might be protected under the proper circumstances. Because of the Report's extended discussion of traditional doctrines that are always applied on a case-by-case basis and CONTU's rejection of proposed alternative limits on protection in favor of such doctrines, the view expressed in Vice-Chairman Nimmer's declaration is probably the better one. Undoubtedly both Congress and CONTU intended the idea-expression and the process-expression dichotomies and the merger doctrine to apply to computer programs, with the effect of denying protection even to literal instructions when necessary to preserve access to ideas and processes. They would probably determine whether structure is protected on a case-by-case basis, consistent with the established principles of copyright law applicable to utilitarian and literary works. Part III develops analytical tools for making this case-by-case determination. However, before doing so, the next section steps back from the structural infringement cases and the legislative history to consider the normative question whether copyright ought to protect program structure.

### C. *Should Program Structure Be Protected by Copyright?*

After reviewing the case law, there can be no doubt that computer program structure *is* protected by copyright. If that protection is narrowly construed, the result is not antithetical to the intentions of Congress, the views of CONTU, or the fundamental purposes of copyright law. This section first explores the purposes of copyright law and the doctrines that limit the scope of copyright protection, including the difficulty of applying limiting doctrines developed in the context of literary works to computer programs. This section next considers the traditional role of patent law in protecting utilitarian works and explores the consequences of giving copyright protection to such works.

117. *See* Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1241 (3d Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987).

Finally, this section concludes that some copyright protection for structure is desirable, provided that it is narrowly tailored to avoid impairing the progress of the computer art.

Copyright protection exists to "promote the Progress of Science and useful Arts by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."[118] Copyright promotes progress by acting as an incentive to authors to create new works and ultimately placing those works in the hands of the public.[119] Copyright has always been characterized by the tension between the need to create a sufficient incentive to create new works and fears of hindering progress by providing too much protection.[120] This tension underlies the selection of a term of protection, although it is impossible to prove that any particular term strikes exactly the right balance between providing an incentive to create and public access.[121]

Doctrines limiting the scope of copyright protection allow courts to strike the proper balance between incentive and impairment in particular cases.[122] The idea-expression dichotomy is the legacy of numerous courts concluding that progress in literature is not well served by allowing the author of a book or play to prevent others from copy-

---

118. U.S. CONST., art. I, § 8, cl. 8; *cf.* A. Bogsch, Inscription on the Cupola of the World Intellectual Property Organization Headquarters Building ("Human genius is the source of all works of art and invention ★ These works are the guarantee of a life worthy of men ★ It is the duty of the states to ensure with diligence the protection of the arts and inventions"), *reprinted in* WORLD INTELLECTUAL PROPERTY ORGANIZATION, GENERAL INFORMATION at i (1988).

119. Sony Corp. of Am. v. Universal City Studios, 464 U.S. 417, 429 (1984) ("The monopoly privileges that Congress may authorize are . . . intended to motivate the creative activity of authors and inventors by the provision of a special reward, and to allow the public access to the products of their genius after the limited period of exclusive control has expired.").

In Mazer v. Stein, 347 U.S. 201 (1954), the Supreme Court noted that

> The economic philosophy behind the clause empowering Congress to grant patents and copyrights is the conviction that encouragement of individual effort by personal gain is the best way to advance public welfare through the talents of authors and inventors in 'Science and useful Arts.' Sacrificial days devoted to such creative activities deserve rewards commensurate with the services rendered.

347 U.S. at 219; *see also* 1 M. NIMMER, *supra* note 40, § 1.03[A].

120. 1 M. NIMMER, *supra* note 40, § 1.05[D]. In Sayre v. Moore, 1 East 361, 102 Eng. Rep. 139 (K.B. 1785), Lord Mansfield noted that:

> [W]e must take care to guard against two extremes equally prejudicial; the one, that men of ability, who have employed their time for the service of the community, may not be deprived of their just merits, and the reward of their ingenuity and labour; the other, that the world may not be deprived of improvements, nor the progress of the arts be retarded.

1 East at 361, 102 Eng. Rep. at 140.

121. Cohen, *Duration,* 24 UCLA L. REV. 1180, 1230-31 (1977) (There is no single correct copyright term; it lies somewhere on a continuum between that which most would agree is too short and too long.); *see also* White, *Why a Seventeen Year Patent?,* 38 J. PAT. OFF. SOCY. 839, 842 (1956) (difficult to justify a patent term on economic grounds).

122. *See* Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 742 (9th Cir. 1971); Goldstein, *Infringement of Copyright in Computer Programs,* 47 U. PITT. L. REV. 1119, 1126 (1986) ("[I]n the copyright lexicon, 'idea' is no more than a metaphor for elements generally belonging in the public domain.").

ing the theme and general plot of the original work.[123] Similarly, the process-expression dichotomy is the legacy of courts holding that while protecting the description of a process appropriately encourages the inventors of processes to share their processes, protecting processes themselves would impair the progress of the useful arts.[124]

Both the idea-expression and process-expression dichotomies limit the scope of copyright protection of the structure of computer programs.[125] Courts deciding cases concerning computer programs have to a greater or lesser extent relied upon cases developing both doctrines. The idea-expression dichotomy is applicable because computer programs are literary works. It should be applied to computer programs as it is applied to other literary works.[126] This is supported by the legal classification of computer programs as literary works,[127] and it is to some extent consistent with the way programmers think of programs.[128]

---

123. *See supra* notes 45-46 and accompanying text.

124. *See supra* notes 53-55 and accompanying text.

125. The Third Circuit explicitly applied the two separate doctrines to computer programs in Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983), *cert. dismissed,* 464 U.S. 1033 (1984). *Compare* 714 F.2d at 1250 (" 'Process', 'System' or 'Method of Operation' ") *with* 714 F.2d at 1252 ("Idea/Expression Dichotomy"). CONTU emphasized the process-expression dichotomy but recognized the application of both doctrines. *See* CONTU REPORT, *supra* note 11, at 20 (discussing the merger doctrine), 25 ("The line which must be drawn is between the expression and the idea, between the writing and the process which is described.").

126. *See supra* notes 114-16 and accompanying text. Applying cases involving novels and plays to computer programs by analogy is not an easy task, since computer programs have no clear counterparts to the plot, theme, or characters of other literary works.

To the extent one can draw analogies to novels and plays, one should analogize a program's function to theme. The *Whelan* court defined "purpose" very broadly. *See supra* note 73. This Note suggests that at the very least, the function of a computer program, *see supra* note 20 and accompanying text, is in every case an idea. This is probably consistent with *Whelan.* The court implied that even programs with similar functions could not be copies of each other absent a showing of at least structural similarity. *See* 797 F.2d at 1238-39. Denying protection to function is a natural result of the axiom that one is always free to program a computer to perform any function that is not patented, CONTU REPORT, *supra* note 11, at 20; *supra* text accompanying note 104, at least as long as one does not copy a protected screen display. *See supra* note 61. Except in the screen display context, one program should never be held to infringe the copyright in another simply by virtue of the fact that it has the same function. If it were, the author of the original program could prevent others from writing better, more efficient programs doing the same thing.

The *Whelan* court correctly identified the "purpose" of a program as an idea, but mistakenly identified purpose and that which is essential to purpose as the only unprotected aspects of the work. *See* 797 F.2d at 1236; *supra* note 72. It is not clear why programs should be held to embody only one idea as a matter of law. While few, if any, programmers have such grand aspirations, nobody would argue that the works of Plato or Shakespeare each express one and only one idea.

127. *See supra* note 1 and accompanying text. However, unlike most other literary works, they have the attribute this Note has called "function." *See* CONTU REPORT, *supra* note 11, at 20-21 ("Programs are a relatively new type of writing . . . . [P]rograms are used in conjunction with machines . . . .").

128. *See* Nagler, 31 COMM. A.C.M. 1034 (1988) (letter to the editor) ("The closest field [to programming] in my opinion is writing. Software is written. . . . Once written, it can be duplicated easily. Some authors write for broad audiences and others write for highly specific . . .

The difficulty with relying upon analogies to literary works is that programs are primarily utilitarian in nature.[129] Although they have some value simply as unambiguous descriptions of solutions to problems, programs usually are written and are important because they can be used to operate computers to achieve some result.[130] For this reason, the process-expression dichotomy is the principal limit upon the copyright protection of computer programs. A computer program clearly is not a process as such: a program is a set of instructions[131] and a process is a series of acts.[132] But a program *is* just as clearly a complete and absolutely unambiguous description of a process. While descriptions of processes are allowed protection under *Baker v. Selden*[133] and section 102(b),[134] that protection cannot extend so far as to allow a copyright owner to control the process he described.[135]

Patent law, not copyright law, provides the traditional mode of protection for utilitarian works such as processes.[136] Processes implemented by computer programs are patentable.[137] The Patent and

---

applications. And, everyone thinks they can write . . . ."). Students of programming "must read and write computer programs — many of them." H. ABELSON & G. SUSSMAN, *supra* note 12, at xi. Commentators have called for "literate programming" and urged programmers to read well-written programs. *See, e.g.*, Bentley, *Literate Programming*, 29 COMM. A.C.M. 364 (1986).

129. *Back to Basics*, *supra* note 52, at 4; Menell, *supra* note 18, at 1046.

130. *See supra* note 11 and accompanying text.

131. 17 U.S.C. § 101 (1988); *see supra* note 11 and accompanying text.

132. *See supra* note 2. At least one court has recognized this distinction. *See* Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1251 (3d Cir. 1983), *cert dismissed*, 464 U.S. 1033 (1984).

133. 101 U.S. 99 (1880).

134. *See supra* notes 42, 52, and accompanying text.

135. *See supra* notes 103-04 and accompanying text.

136. Goldstein, *supra* note 122 at 1122-23; Samuelson, *supra* note 58, at 732-36.

137. *See* Lundberg, Sumner & Michel, *Twelve Myths About Patent Protection for Software*, BULL. L. SCI. & TECH., Nov./Dec. 1988, at 2 [hereinafter *Twelve Myths*]; Smith, Yoches & Anzalone, *Computer Program Patents*, COMPUTER LAW., Apr. 1988, at 1, 2, 7. To receive a patent, an inventor must invent or discover a "*new* and *useful* process, machine, manufacture, or composition of matter, or any new and useful improvement thereof." 35 U.S.C. § 101 (1988) (emphasis added); *see also* 1 D. CHISUM, PATENTS § 1.01 (1989). Computer programs are not themselves patentable since computer programs do not satisfy any of the subject matter categories of 35 U.S.C. § 101 (1988).

To be patentable, a process must meet the tests of novelty (*i.e.*, that particular invention has never before been invented), 35 U.S.C. § 102 (1988); 1 D. CHISUM, *supra*, § 3.01, and nonobviousness (*i.e.*, the invention would not have been obvious to a person of ordinary skill in the art of the invention in light of all prior developments in that art). 35 U.S.C. § 103 (1988); 2 D. CHISUM, *supra*, § 5.01.

Laws of nature, including mathematical formulae and "algorithms," are not patentable. However, inventions employing formulae and algorithms are patentable if there is more to the invention than just a formula or algorithm. *See* Diamond v. Diehr, 450 U.S. 175 (1981) (manufacturing process employing well-known equation and a programmed computer is patentable); Gottschalk v. Benson, 409 U.S. 63 (1972) (method of converting binary coded decimal numbers to binary numbers is not patentable); *In re* Pardo, 684 F.2d 912, 915-16 (C.C.P.A. 1982) (method of executing programs in a computer is patentable); *In re* Abele, 684 F.2d 902, 905-06 (C.C.P.A. 1982) (invention not patentable if nothing more than an algorithm is claimed; otherwise allowing

Trademark Office ("PTO") has issued a large number of patents claiming processes implemented by computer programs.[138] The owner of a patent has the right to exclude all others from making, using, or selling the patented invention in the United States[139] for a term of seventeen years.[140] The PTO must examine a patent application before it will issue a patent.[141] The creator of a patentable invention thus receives greater protection for a shorter period of time and with greater difficulty than the creator of a work protected by copyright.[142]

Although both copyright and patent law are designed to provide a significant incentive to create and disseminate new works, the different

---

the claim would grant a monopoly over the algorithm); *In re* Taner, 681 F.2d 787, 790 (C.C.P.A. 1982) (method of seismic prospecting involving the "summing" of certain electrical signals is patentable); *see also* PATENT AND TRADEMARK OFFICE, U.S. DEPT. OF COMMERCE, MANUAL OF PATENT EXAMINING PROCEDURE § 2106 (5th ed. rev. 1989) (describing patent examining procedures for inventions involving computer programs). *But see* Parker v. Flook, 437 U.S. 584 (1978) (use of formula to update alarm limits in manufacturing process unpatentable since formula part of prior art and remainder of invention obvious in light thereof). *See generally* 1 D. CHISUM, *supra*, § 1.03[6]; Chisum, *The Patentability of Algorithms*, 47 U. PITT. L. REV. 959 (1986); Gemignani, *Should Algorithms Be Patentable?*, 22 JURIMETRICS J. 326 (1982).

138. *See Twelve Myths*, *supra* note 137, at 2; Bulkeley, *Will Software Patents Cramp Creativity?*, Wall St. J., Mar 4, 1989, at B1, col. 3. Examples of patents claiming processes implemented by computer programs include: U.S. Patent No. 4,374,408 (Feb. 15, 1983) (system and method for translating programs in the RPG programming language to COBOL); U.S. Patent No. 4,355,371 (Oct. 19, 1982) (method for automatic spelling error correction); U.S. Patent No. 4,309,756 (Jan. 5, 1982) (method for compiling a program). Chisum, *supra* note 137, at 1021-22 lists a number of patents involving computer programming techniques.

While otherwise patentable processes implemented by computer programs are patentable, most programs are probably not patentable. 1 D. BENDER, COMPUTER LAW § 3A.02, at 3A-6.1 (1988) (over 90% of programs not patentable); Davidson, *Protecting Computer Software: A Comprehensive Analysis*, 23 JURIMETRICS J. 339, 357 (1983) (perhaps less than 1% of programs patentable).

139. 35 U.S.C. § 271 (1988).

140. 35 U.S.C. § 154 (1988).

141. 35 U.S.C. § 131 (1988); 3 D. CHISUM, *supra* note 137, § 11.03.

142. To receive a patent, an inventor must demonstrate that his invention is useful, novel, and nonobvious. *See, supra* note 137. A patent application is subject to numerous statutory requirements. *See* 35 U.S.C. §§ 111-115 (1988); 3 D. CHISUM, *supra* note 137, § 11.02. Perhaps the most important requirement is that the application must disclose the best known mode of making and using the invention in terms that enable one of average skill in the relevant art to make and use it. 35 U.S.C. § 112 (1988); *see also* 2 D. CHISUM, *supra* note 137, ch. 7. Since the PTO cannot issue a patent until an examination reveals that the inventor is entitled to a patent, it generally takes from two to three years from the time of filing to obtain a computer program patent. *See Twelve Myths*, *supra* note 137, at 3 (two years); Stern, *supra* note 9, at 1247 n.92 (three years). Including patent application, issuance, and attorney's fees, the cost of obtaining a patent for a computer program might in some cases reach $10,000. *Twelve Myths*, *supra* note 137, at 4; *see also* Stern, *supra* note 9, at 1247 n.92 (usually over $5000). Patents are also subject to the payment of maintenance fees. 35 U.S.C. § 41(b) (1988); 3 D. CHISUM, *supra* note 137, § 11.02[1][d].

Copyright, on the other hand, subsists in works from the moment they are fixed tangibly. Compliance with the 1976 Act's relatively few statutory formalities is easy and inexpensive. *See supra* note 34. Since the independent creation and the exploitation of a substantially similar work do not constitute copyright infringement, *see supra* note 38, a copyright grants much narrower protection than a patent. Patent, copyright, and trade secret law are compared in general terms in tabular form in CONTU REPORT, *supra* note 11, at 19.

conditions of protection reflect a belief that the exclusive right to prevent others from using a useful process is a powerful right that should be granted only for relatively short periods and under carefully circumscribed conditions. This belief motivates the process-expression dichotomy, which defines the boundary between copyright and patent law by permitting copyright protection of descriptions of processes while denying protection to the processes themselves (which are, under the proper circumstances, patentable). Failure adequately to consider process-expression merger when faced with a case of alleged structural infringement risks withholding useful processes from the public domain without the safeguards of the patent system.

Progress in all fields requires building upon the works of others.[143] Progress in the computer art is no exception. Ideas for program functions, structures, and low-level implementations may all come from the work of others and then be improved upon or combined with original material.[144] The computer art advances because programmers improve upon the functions of existing programs and the processes they embody and then write new programs, possibly employing preexisting implementation details from the public domain.

Both patent and copyright law limit a programmer's ability to build upon the work of others. Some limitation is clearly necessary to provide an incentive to create new works. If any or all aspects of a new program could be copied with impunity, there would be much less incentive to create new programs and most programs would be distributed only in secrecy. This would deprive the public of the benefit of most innovations and slow the progress of the art. Granting extensive protection might allow knowledge of innovations, but would withhold them from the public domain for long periods and thus also slow the progress of the art.

Protecting processes would be a most injurious extension of copyright protection for several reasons. First, processes are utilitarian. Since they are often capable of use in various contexts, the effects of protecting processes would be especially harsh. Second, protected processes would not be examined to determine whether they were worthy of the societal cost of protection. Third, the seventy-five year copyright term is a very long time to deprive the public of a new and useful process. Finally, protected processes could not intentionally be

143. In Emerson v. Davis, 8 F. Cas. 615 (C.C.D. Mass. 1845) (No. 41,436), Justice Story noted that:

> In truth, in literature, in science and in art, there are, and can be, few, if any, things, which, in an abstract sense, are strictly new and original throughout. Every book in literature, science and art, borrows, and must necessarily borrow, and use much which was well known and used before.

8 F. Cas. at 619.

144. *See supra* notes 28-29 and accompanying text.

created independently.[145]

To deny all protection of structure would theoretically reduce the incentives to create new, innovative structures. This reduction would, however, be small, because the desire for efficient implementations of program functions is a powerful incentive to create new structures.

A more important reason for protecting structure is that this protection is necessary to give meaning to the protection of literal instructions. If structure were not protected at all, a translation or a program copied and extensively but trivially modified would not clearly infringe the copyright in the original program. This was an important factor in early decisions to extend copyright protection to the plots of plays.[146] However, protecting structure in its entirety runs the risk of protecting the underlying ideas and processes. Structure should be protected, but only to the extent that protection does not limit access to ideas or processes.

On balance, it appears that structure should not be excluded from the realm of copyright protection as a matter of law. The courts have not done so. Although the intentions of Congress and CONTU are not entirely clear, it does appear that they would consider protection of structure on a case-by-case basis by applying traditional doctrines. Some protection of structure, under limited circumstances, is necessary to best promote progress in the computer art. Part III develops variations on traditional copyright doctrines tailored to the specific situation of computer programs. These approaches attempt to strike the proper balance between granting incentives to create and impairing progress in the computer art. This achieves the fundamental purpose of copyright law, promoting the progress of science and the useful arts.

## III. Determining When the Copyright in a Computer Program Has Been Infringed by the Copying of Its Structure

Part II of this Note concluded that the structure of a computer program may, under the proper circumstances, constitute protected expression and should therefore receive copyright protection. This Part develops approaches to determine if, and to what extent, the structure of any particular program should receive protection. These approaches are nothing revolutionary; indeed, they are simply the levels of abstraction test and the merger doctrine recast in the terms of

145. It would be impossible to use a clean room technique intentionally to create a protected process independently. Protecting the process leaves little unprotected material to distill out of the program for use as a guide to writing a new program in a clean room.

146. *See, e.g.*, Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930) (copyright protection "cannot be limited literally to the text, else a plagiarist would escape by immaterial variations"), *cert denied*, 282 U.S. 902 (1931).

the computer art. In contrast, some commentators considering the protection of computer program structure have relied almost exclusively upon the policy underpinnings of the idea-expression and process-expression dichotomies to arrive at approaches for distinguishing idea and process from expression that are only loosely tied to traditional doctrines. Others have misapplied traditional doctrines. This Part roots its approaches to applying the idea-expression and process-expression dichotomies firmly in traditional doctrines and then places the analysis applicable to these dichotomies in the context of a conventional infringement action inquiry into substantial similarity.

### A. *Distinguishing Idea from Expression in the Context of Program Structure*

Although Learned Hand wrote that the distinction between idea and expression must "inevitably be *ad hoc*,"[147] courts and commentators have proposed several helpful approaches to this problem. Judge Hand did so when he proposed his levels of abstraction test.[148] Professor Chaffee did so with his pattern test.[149] The *Whelan* court did so when it developed its own test.[150] Those who believe program structure should never be protected must base their belief upon some means of distinguishing idea from expression.

Part II observed that doctrines applicable to other literary works should guide the search for the dividing line between idea and expression in the case of computer programs. That observation necessarily implicates the levels of abstraction approach.[151] Judge Hand built his abstractions upon the whole work, and so it is with the whole of the program that one must begin. Just as the plot of a play provides a framework for building abstractions upon the play, the structure of a computer program provides a framework for building abstractions upon the program. At the lowest level of abstraction, a computer program may be thought of in its entirety as a set of individual instructions organized into a hierarchy of modules. At a higher level of abstraction, the instructions in the lowest-level modules may be replaced conceptually by the functions of those modules. At progressively higher levels of abstraction, the functions of higher-level modules conceptually replace the implementations of those modules in terms of lower-level modules and instructions, until finally, one is left with nothing but the ultimate function of the program.[152] This pro-

147. Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960); *see supra* note 49 and accompanying text.

148. *See supra* notes 47-49 and accompanying text.

149. *See supra* note 50 and accompanying text.

150. *See supra* note 72 and accompanying text.

151. *Silicon Epics, supra* note 12, at 1568-69.

152. The *Whelan* court's "purpose" is a sort of metafunction of the program — it extends

cess of building abstractions is in some sense the opposite of the process that the programmer probably used in designing the program.[153]

At any intermediate level of abstraction, the program may be conceived of as a collection of module functions and literal instructions organized into higher-level modules that together implement the ultimate function of the program. At higher levels of abstraction, the module functions are relatively more general, the structure is relatively more simple, and the instructions that provide the "glue" to hold the modules together are, in general, fewer in number. A program has structure at every level of abstraction at which it is viewed. At low levels of abstraction, a program's structure may be quite complex; at the highest level it is trivial. Under this approach, text and structure are treated as an integrated whole. Analytically, then, literal copying

---

beyond the program itself to the end it serves for the user — so it may be thought of as a level of abstraction that is more general still.

Two commentators have suggested an example of the application of levels of abstraction analysis to the *structure* of computer programs rather than the programs themselves:

> [A]n "abstraction" analysis might define structure at one level of abstraction by a detailed flowchart that reflects the operation of each subroutine of the program as well as the relationships among subroutines. At a somewhat higher level of abstraction, structure might be depicted by a flowchart of the logical links among subroutines. At an even higher level, still possibly considered to depict structure, the program could be described by a simple list of the subroutines in the order they appear (e.g., "billing routine followed by accounts payable routine").

Ladd & Joseph, *supra* note 70, at 10. The example begins much as the approach suggested in the text by identifying the implementation of the program at its lowest level as the lowest level of abstraction. The authors next move to a representation of the whole structure of the program and finally to organization wholly independent of structure or function. These levels correspond roughly to what this Note has called sequence, structure, and organization, respectively. *See supra* note 27 and accompanying text. This approach is easy to apply and a step in the right direction, but it treats the structure of the program as something independent of the program and builds its abstractions upon the structure of the program rather than the program itself. This considers insufficiently the possibility that lower levels of structure might contain more protected expression than higher levels. It is unclear what questions one must ask to determine where in Ladd and Joseph's hierarchy of levels the division between idea and expression falls. It is also unclear what it means to say, for example, that structure is protected as such but organization is not and whether that could possibly be a correct result.

153. *See supra* notes 19-25 and accompanying text. Several commentators have briefly described a similar approach with greater emphasis upon the relationship between the levels of abstraction and the programmer's approach to writing the program. Reback & Siegel, *Toward a Comprehensive Test for Software Copyright Infringement,* COMPUTER LAW., Dec. 1984, at 1, 4-10; *see also* Reback & Hayes, *The* Plains *Truth: Program Structure, Input Formats and Other Factual Works,* COMPUTER LAW., Mar. 1987, at 1, 4-7. One difference between the approach described by these commentators and that described here is that they seemingly equate levels of abstraction with levels of a program's structure. *See* Reback & Hayes, *supra,* at 6; Reback & Siegel, *supra,* at 4. This Note would instead view a program from a very large number of levels of abstraction. At each level, the program has a structure. This Note forms these abstractions by conceptually substituting the functions of modules for their implementations at different levels of a program's structure and in different parts of a program. While Reback, Hayes, and Siegel would identify a single level of a program's structure as a "level of abstraction," this Note suggests that one abstract view of a program might consist of a particular level of that program's structure, the literal instructions and the functions of all of the modules at that level of the program's structure, and all of the higher levels of the program's structure.

is simply copying parts of the lower levels, and if the copying is extensive, all levels of a program's structure.

Developing a set of abstract views of a program is relatively easy; the task of determining the point in this set of abstractions at which the additional detail in the next-lowest level constitutes expression rather than idea is much more difficult. Professor Chaffee's pattern test[154] is an attempt at a general solution to this problem in the context of novels and plays. By its own terms, it is not well suited to computer programs. Whether there is any expression in the structure of a program viewed at a particular level of abstraction can only be answered after a more fundamental inquiry into the details of the program.

Beginning with the highest level of abstraction, the function of a program is in every case an idea.[155] At any lower level of abstraction, one must consider the details present in the structure of the program at that level of abstraction but missing from the structure of the program at the next highest level of abstraction to determine whether they too are idea. These details consist of a set of module functions and client relationships.[156] At each level, these details should be considered part of the idea of the program if they are not sufficiently original to be protected or the use of these structural details is necessary[157] efficiently to implement the program as it exists at the higher level of abstraction.[158] Eventually, one will reach the level of individual instructions. Even sequences of literal instructions cannot constitute protected expression when they are not original or they are necessary to the efficient implementation of the program's function.[159]

---

154. *See supra* text accommpanying note 50.

155. *See supra* note 126.

156. *See supra* note 27 and accompanying text.

157. Although a program could always theoretically be implemented with a trivial structure, this could prove quite difficult and would be contrary to general programming practices. *See supra* note 21 and accompanying text. No aspect of any program's structure should be found protected on the grounds that the underlying function could conceivably be implemented in a way contrary to generally accepted programming practices.

158. *See* 3 M. NIMMER, *supra* note 40, § 13.03[A][1]; Reback & Siegel, *supra* note 153, at 4-5. This inquiry is in some sense the same inquiry made in Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1238 (3d Cir. 1986) ("[T]he structure of the program was not essential to that task . . . ."), *cert. denied,* 479 U.S. 1031 (1987), and Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1253 (3d Cir. 1983), *cert dismissed,* 464 U.S. 1033 (1984), except here it is conducted level by level and incorporates an efficiency qualification. The approach to distinguishing idea from expression suggested here differs from the *Whelan* rule in its definition of function. *See supra* notes 20, 73, and accompanying text. Moreover, the *Whelan* rule lacks an explicit means of distinguishing process from expression such as that proposed in section III.B. This Note's approach is also in some respects similar to that described in Davidson, *supra* note 38, at 1082-85. Davidson does not explicitly build levels of abstraction but rather conceives of a program as a collection of nested "black boxes," which correspond to modules, and then determines a level of protection for each black box.

159. CONTU REPORT, *supra* note 11, at 20. No single instruction can ever be very original; neither can any individual word of a novel or play. The question is whether there is sufficient originality in some meaningful sequence of instructions. *See* Declaration of Melville B. Nimmer, *supra* note 114, ¶¶ 14-15. For example, a well-known and often used sequence of instructions for

Although this inquiry is by design essentially the same as that applicable to any literary work, it is necessary to make one concession to the utilitarian nature of computer programs. Computational efficiency, which might be defined differently depending upon one's objectives, plays a role in this inquiry because the use of some structures over others may be indicated by concern for the efficient use of computer or human resources.[160] If ideas and, in particular, functions are not to be protected by copyright, surely it is no response to a claim that the use of a particular set of modules is necessary to implement a function to argue that the accused program might have been written in a way that requires larger amounts of memory space or processor or programmer time, or makes the program more difficult to learn or use.

While it is absolutely essential that necessary or conventional implementations of a function not be protected, there will probably be relatively few cases where a particular implementation of a function is necessary. With the possible broad exception of operating system programs, where the design of the computer may actually dictate some significant aspects of the implementation of the program,[161] there are usually many ways to implement any function.[162] For example, even if all teleprompter programs would have the same four modules at the highest level,[163] the likelihood that any of these modules could be implemented efficiently with only one structure is probably very small. Claims of necessity should thus be viewed with some skepticism.

This method of distinguishing idea from expression in the context of computer program structure, regrettably, may not be very easy to apply. Distinguishing idea from expression is never easy, and the computer program context makes this inquiry especially difficult for

---

sorting the items in a list is no more deserving of copyright protection than a dozen lines of *Hamlet* contained in an otherwise original play.

160. One commentator recently suggested that copyright should not protect aspects of a structure motivated by such functional concerns as efficiency of execution. Menell, *supra* note 18, at 1085. This concern for not protecting aspects of structures resulting from design decisions made in the interest of computational efficiency stands behind the use of efficiency as a qualification in the approaches proposed in this Note. This Note stops just short of the position advocated by Professor Menell. Both this Note and Menell would require a defendant to explain why the plaintiff's structural design decisions best met the defendant's efficiency objectives. *Id.* at 1086. Menell, however, would require the plaintiff to prove that the plaintiff himself made inefficient design decisions. *Id.* at 1086-87. This Note would not require such proof and would not find idea-expression merger (or process-expression merger) in the cases where other structures could be used to implement the function (or process) in accordance with the defendant's efficiency objectives.

161. *See Franklin,* 714 F.2d at 1253.

162. CONTU questioned a Vice-President of the Association for Computing Machinery about this point. He stated that there are in practice dozens or even hundreds of ways to implement most functions. CONTU REPORT, *supra* note 11, at 20 n.106; *see supra* note 24 and accompanying text.

163. This was a finding in Q-Co Indus., Inc. v. Hoffman, 625 F. Supp. 608, 616 (S.D.N.Y. 1985); *see supra* note 89. This suggests that the implementation of the function of a teleprompter program with these modules is both unoriginal and necessary.

finders of fact. Even though this approach effectively requires examining successive levels of modules as they might be represented in a structure diagram,[164] this is not quite as simple as sliding a ruler down a page. The determinations to be made require a firm grasp of the program in question and some understanding of the computer art.[165] Some modules of a program may be more original or less necessary to the implementation of other, higher-level modules than others. Thus, possibly not even the literal instructions implementing one high-level module of a program would constitute protected expression while all of the structure of a second high-level module might.

In the ordinary case, part of the structure of a program will be considered idea and part will be considered protected expression. While the possibility that only certain aspects of the plot of a play may constitute protected expression is not at all controversial, courts have unanimously treated the protection of a program's structure as an all-or-nothing proposition. There is no legal basis for such an approach, although the facts of a particular case might support a decision that all or none of the structure of a program is protected. While determining which aspects of the structure of a program are protected is much more difficult than simply saying that all or none of a program's structure is protected, courts must be prepared to make this inquiry in order best to effectuate the fundamental purposes of copyright law.

Because this approach for distinguishing idea from expression is simply a special case of an approach applicable to all literary works, it is a relatively weak means for ensuring that the public has access to all of the aspects of a computer program to which the public should have access. Computer programs are utilitarian works, and this approach does nothing to ensure public access to processes. It ensures public access to functions, but only high levels of programs' structures will often be necessary to the efficient implementation of functions. The next section develops a different and stronger, yet complementary, approach based upon the process-expression dichotomy.

### B. *Distinguishing Process from Expression in the Context of Computer Program Structure*

Because computer programs are utilitarian works, the process-expression dichotomy is an important limit on the scope of their protection. The process-expression dichotomy was the primary focus of CONTU's discussion of the scope of copyright protection of computer

164. This is true at least when it is possible to represent a program's structure in the form of a simple diagram patterned after an organizational chart. There are a number of reasons why the structure of a program might not be amenable to such simple representation or analysis. *See supra* note 27. In those cases, some modification of the basic levels of abstraction approach that remains true to the ultimate purpose of the approach should be employed.

165. Without expert testimony to guide the finder of fact, the approach is impossible to use. *See infra* notes 186-96.

programs. In spite of this, courts have generally not explicitly recognized the process-expression dichotomy as a limit on the protection of computer programs independent of the idea-expression dichotomy. Since few other works protected by copyright pose process-expression merger problems, there is no well established test for distinguishing protected descriptions of processes from descriptions that are merged with their processes. *Baker v. Selden*[166] is of little or no help, for Selden argued that the process described was protected and the Court simply concluded that it was not.[167] The task here is analogous to determining whether the division of Selden's essay into sections and paragraphs was protected by copyright or merged with his unprotected accounting method.[168]

To make this determination in a refined way, it is necessary to examine each of the levels of modules in a program. Because this excrimination involves no abstract views of the program, it is not, strictly speaking, a levels-of-abstraction approach. It could, however, be applied conveniently in tandem with an idea-expression inquiry using a levels-of-abstraction approach.

Because the ultimate function of a program is an idea, the starting place of this examination is the set of modules at the next highest level of a program's structure. The process embodied in a program is the sequence of acts it instructs a computer to perform.[169] To ensure that copyright protection of a program's structure does not extend to its process, one must individually consider this highest-level set of modules and all other sets that consist of modules sharing a common client module. For each set of modules, one should ask whether the use of *this particular set* of modules is necessary efficiently to implement that part of the program's process that is implemented in the common client module. If the answer is affirmative, the common client's use of the lower-level module into modules is merged with the process of the

166. 101 U.S. 99 (1880).

167. *See supra* text accompanying notes 53-55.

168. A useful question to keep in mind while considering whether a description of a process is protected by copyright is whether that description should be allowed into a clean room, *see supra* note 38, in which another description of the process is being written. If copyright protection of computer programs is to have any meaning at all, a whole program, complete with literal instructions, should probably never be allowed into the clean room in which a competing program is being written. On the other hand, a statement of the function of the program would always be allowed into the clean room.

169. CONTU understood the unprotected process described in a computer program to be the sending of electrical impulses through the computer. CONTU REPORT, *supra* note 11, at 22. This understanding is, in some respects, troubling because a particular set of electrical signals can be sent through a computer by one and only one program in that computer's computer language. *See supra* notes 12-13 and accompanying text. The courts have nonetheless had little difficulty in finding programs written in computer languages protected by copyright. *See supra* note 58. Since electrical signals in computers produce certain results, CONTU's understanding of the nature of an unprotected process can safely be thought of as rendering a series of acts of a slightly more general nature, such as adding together two specified numbers or even calculating the value of some complex formula, a process outside the scope of copyright protection.

program and is unprotected. While the idea-expression inquiry in section III.A. will tend to find idea-expression merger at high levels of a program's structure, this approach will tend to find process-expression merger at intermediate and low levels of a program's structure.

One could easily be overzealous in finding process-expression merger when making this inquiry. If the same process could be implemented efficiently by combining and redividing the functions of a set of modules in a number of substantially different ways, then the selection of a particular set of modules should be protected. This is especially true at low levels of a program's structure where there may be a significant amount of choice in allocating acts among modules.

This inquiry guards against the monopolization of a process by the enforcement of a copyright in a program describing that process.[170] In the abstract, it is unclear how much of a typical program's structure this inquiry would remove from the realm of expression. Because many low-level structural design decisions motivated by a concern for computational efficiency will probably be found merged with process, however, it is probably much greater than that removed by the mere application of the levels of abstraction approach for distinguishing idea from expression. Elements of structure identified as either idea or process are not protected by copyright. The two approaches together lead to a narrower scope of protection than has been found by most courts and proposed by many commentators. A relatively narrow scope of protection is, however, necessary to prevent monopolization of the ideas and processes embodied in computer programs. Monopolization impairs progress in the computer art by severely limiting the opportunities for programmers to make incremental improvements upon the ideas and processes described by others.

### C. *The Place of Idea-Expression and Process-Expression Inquiries in a Conventional Infringement Analysis*

Distinguishing among idea, process, and protected expression is a necessary and important part of deciding cases involving the alleged infringement of the structure of computer programs. This section discusses the application of the approaches developed above in such cases. A typical case of this genre probably involves two fairly large commercial programs written in different languages.[171] The alleged copy will have some higher-level modules with functions that are simi-

170. Returning briefly to the hypothetical clean room, *see supra* note 168, any aspect of a program's structure that is, or is merged with, an idea or process should be allowed into the clean room.

171. *See* Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1226 (3d Cir. 1986) (EDL and BASIC), *cert. denied*, 479 U.S. 1031 (1987); Q-Co Indus., Inc. v. Hoffman, 625 F. Supp. 608, 614 (S.D.N.Y. 1985) (Pascal and BASIC).

lar to those of modules in the original program,[172] and perhaps a few short sequences of instructions that are approximately direct translations of corresponding instructions in the original.[173] The programs may also have similar screen displays,[174] employ similar data structures,[175] or otherwise resemble one another. The alleged copy may have been written by former employees of the plaintiff,[176] while the defendant possessed a copy of the plaintiff's program under a license,[177] or while the plaintiff's program was available commercially.[178] The court must determine whether the defendant's program infringes the plaintiff's copyright.

Recall that the plaintiff must prove ownership of the copyright in the program and copying by the defendant.[179] Since it is seldom possi-

---

172. *See Q-Co,* 625 F. Supp. at 614 (4 similar high-level modules); E.F. Johnson Co. v. Uniden Corp. of Am., 623 F. Supp. 1485, 1496-97 (D. Minn. 1985) (38 similar modules).

173. *See Q-Co,* 625 F. Supp. at 614 (a few textual similarities); *E.F. Johnson,* 623 F. Supp. at 1495-96 (describing errors and idiosyncracies common to both programs); SAS Inst., Inc. v. S & H Computer Sys., Inc., 605 F. Supp. 816, 822 (M.D. Tenn. 1985) (44 instances of direct copying of literal text).

Section 106(2) of the 1976 Act grants a copyright owner the exclusive right "to prepare derivative works based upon the copyrighted work." 17 U.S.C. § 106(2) (1988). This includes the right to translate the work from one language to another. *See* 2 M. NIMMER, *supra* note 40, § 8.09[B][1]. The same idea-expression and process-expression problems applicable to computer programs generally are applicable to those works with logic so similar that the possibility of direct translation might be raised in litigation. However, one court, in dicta, has opined:

> [I]t is as clear an infringement to translate a computer program from, for example, FORTRAN to ALGOL, as it is to translate a novel or play from English to French. In each case the substance of the expression (if one may speak in such contradictory terms) is the same between original and copy, with only the external manifestation of the expression changing. Likewise, it would probably be a violation to take a detailed description of a particular problem solution, such as a flowchart or step-by-step set of prose instructions, written in human language, and program such a description in computer language.

Synercom Technology, Inc. v. University Computing Co., 462 F. Supp. 1003, 1013 n.5 (N.D. Tex. 1978). A case perhaps addressing the point raised in the last sentence of the quoted passage is Williams v. Arndt, 626 F. Supp. 571 (D. Mass. 1985). There, the court held that the copyright in a book describing "a detailed, step-by-step procedure or process" for commodity trading was infringed by a computer program based upon that book. 626 F. Supp. at 578. Although one may sympathize with the plaintiff, it is difficult to ignore the similarities between this case and *Baker,* a point that the court ignores. The *Synercom* dicta is not necessarily incorrect; instead, it should be limited to descriptions so detailed that producing a computer program from the description is a mechanical task. One must also be careful that literal instructions merged with the idea or process components of the program are not protected.

174. *See* Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 609 F. Supp. 1307, 1322 (E.D. Pa. 1985), *affd.,* 797 F.2d 1222 (3d Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987).

175. *See Whelan,* 797 F.2d at 1228; *E.F. Johnson,* 623 F. Supp. at 1490, 1494.

176. *See* Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1174 (9th Cir. 1989); Plains Cotton Co-op. Assn. v. Goodpasture Computer Serv., Inc., 807 F.2d 1256, 1258-59 (5th Cir.), *cert. denied,* 484 U.S. 821 (1987); *Q-Co,* 625 F. Supp. at 611-13.

177. *See Whelan,* 797 F.2d at 1226-27; SAS Inst., Inc. v. S & H Computer Sys., Inc., 605 F. Supp. 816, 821 (M.D. Tenn. 1985).

178. *See* Pearl Sys. v. Competition Elecs., Inc., 8 U.S.P.Q.2d (BNA) 1520, 1521 (S.D. Fla. 1988) (program embodied in memory chip in shot timer advertised in magazines); E.F. Johnson Co. v. Uniden Corp. of Am., 623 F. Supp. 1485, 1488-89 (D. Minn. 1985) (program in memory chip in radio sold by 150 authorized dealers).

179. *See supra* notes 40-41 and accompanying text.

ble to offer direct proof of copying,[180] the law permits an inference of copying from substantial similarity between the works and the defendant's access to the plaintiff's work.[181] Once a plaintiff has established a prima facie case of access and substantial similarity, the defendant bears the burden of proof on defense of independent creation.[182] If access cannot be proved directly, the plaintiff still may succeed if the similarities are "so striking as to preclude the possibility that plaintiff and defendant independently arrived at the same result."[183] To date, it has not been difficult to prove access in copyright cases involving computer program structure.[184] The determination of substantial similarity, however, is often very difficult.[185]

Under the rule of *Arnstein v. Porter,*[186] substantial similarity should be considered in two parts: (1) whether defendant copied from plaintiff; and (2) if so, whether the works were so similar that the copying constituted an unlawful appropriation. The first question may be decided with the aid of expert testimony analyzing the works in question and pointing out specific similarities.[187] The question is whether there was copying at all. The trier of fact need not worry

180. To prove copying directly requires a witness to the act of copying. If the copying was subconscious, *see generally* Harold Lloyd Corp. v. Witwer, 65 F.2d 1, 16 (9th Cir. 1933), there could be no witnesses, since there was no act outwardly recognizable as copying. 3 M. NIMMER, *supra* note 40, § 13.01[B]; *see also* Warner Bros., Inc. v. American Broadcasting Cos., 654 F.2d 204, 207 (2d Cir. 1981).

181. *Warner Bros.*, 654 F.2d at 207; Ferguson v. National Broadcasting Co., 584 F.2d 111, 113 (5th Cir. 1978); Sid & Marty Krofft Television Prods. v. McDonalds Corp., 562 F.2d 1157, 1162 (9th Cir. 1977); Reyher v. Children's Television Workshop, 533 F.2d 87, 90 (2d Cir.), *cert. denied*, 429 U.S. 980 (1976); 3 M. NIMMER, *supra* note 40, § 13.01[B].

182. 3 M. NIMMER, *supra* note 40, § 12.11[D].

183. Arnstein v. Porter, 154 F.2d 464, 468 (2d. Cir. 1946), *cert. denied,* 330 U.S. 851 (1947); *accord* Baxter v. MCA, Inc., 812 F.2d 421, 423 (9th Cir. 1987); 3 M. NIMMER, *supra* note 40, § 13.02[B]. Similarity should never be found "striking" unless there is a substantial amount of literal copying.

184. *See, e.g.,* Plains Cotton Co-op. Assn. v. Goodpasture Computer Serv., 807 F.2d 1256, 1258-59 (5th Cir.) (same programmers wrote both programs and had access to designs of first when writing second), *cert. denied,* 484 U.S. 82 (1987); Whelan Assocs. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1232 (3d Cir. 1986) (defendant used plaintiff's program and was sales representative for plaintiff), *cert. denied,* 479 U.S. 1031 (1987); SAS Inst., Inc. v. S & H Computer Sys., Inc., 605 F. Supp. 816, 821 (M.D. Tenn. 1985) (defendant had license for much of plaintiff's source code).

185. *Warner Bros.*, 654 F.2d at 208 (quoting 3 M. NIMMER, *supra* note 40, § 13.03[A]).

186. 154 F.2d 464 (2d Cir. 1946), *cert. denied,* 330 U.S. 851 (1947). In Sid & Marty Krofft Television Prods. v. McDonald's Corp., 562 F.2d 1157 (9th Cir. 1977), the Ninth Circuit announced a somewhat different rule permitting greater use of expert testimony, Note, *supra* note 10, at 510 n.76, and expanding the role of the trier of fact, 3 M. NIMMER, *supra* note 40, § 13.03[E]. Professor Nimmer suggests that the *Krofft* rule is based on an erroneous reading of *Arnstein,* has not been followed in some subsequent Ninth Circuit cases, and should not be followed in the future. *Id.* This Note is primarily concerned with the predominant *Arnstein* rule.

187. *Arnstein,* 154 F.2d at 468 (dissection and expert testimony allowed on question of copying); *see also* Walker v. Time Life Films, Inc., 784 F.2d 44, 51 (2d Cir.), *cert. denied,* 476 U.S. 1159 (1986) (allowing consideration of affidavit of expert analyzing "plot, themes, structure, characters, and pace of both works"); Scott v. WKJG, Inc., 376 F.2d 467, 469 (7th Cir.) (expert testimony on similarities between plays admissible), *cert. denied,* 389 U.S. 832 (1967); Midway

about deciding whether protected expression was copied.[188] Only if there was copying is it necessary to reach the second question, whether there was an unlawful appropriation. Here, the trier of fact must determine whether the similarity between the works includes substantial protected aspects of the plaintiff's work.[189]

The answer to the second question should be based upon the spontaneous impression of an ordinary observer.[190] Although the *Arnstein* court noted that " 'dissection' and expert testimony are irrelevant"[191] to this second question, *Arnstein* did not involve application of the idea-expression and process-expression dichotomies.[192] Some courts have indicated that the trier of fact is to compare only the protected portions of a plaintiff's work with the defendant's work,[193] and others have explicitly found that the *Arnstein* test allows the trier of fact to dissect the works to determine what is protected.[194] In cases involving idea-expression and process-expression questions, the trier of fact should answer the second of the *Arnstein* questions by determining which aspects of the plaintiff's work are protected and then deciding whether there was unlawful appropriation of protected expression.[195]

The approaches developed in the first two sections of this Part are employed to determine which parts of the structure of the plaintiff's program are merged with an idea or a process and are thus unprotected. These approaches involve dissecting a plaintiff's program in the classic copyright sense, a detailed examination to determine which parts are protected and which are not. A layperson could not employ

---

Mfg. Co. v. Bandai-America, Inc., 546 F. Supp. 125, 138 (D.N.J. 1982) (same); 3 M. NIMMER, *supra* note 40, § 13.03[E].

188. 3 M. NIMMER, *supra* note 40, § 13.03[E], at 13-62.11 to 13-62.12.

189. *Id.* at 13-62.12 to 13-62.13.

190. *Arnstein,* 154 F.2d at 468. For a discussion of who the ordinary observer of a computer program ought to be, see *Silicon Epics, supra* note 12, at 1571-73.

191. *Arnstein,* 154 F.2d at 468.

192. *Arnstein* involved the alleged infringement by Cole Porter of the copyrights in several of the plaintiff's songs. 154 F.2d at 467. The court may not have carefully considered the application of its rule to cases involving difficult idea-expression or even process-expression questions.

193. *See* Millworth Converting Corp. v. Slifka, 276 F.2d 443, 445 (2d Cir. 1960) (enumerated elements of plaintiff's expression not copied, but rather general ideas); Morse v. Fields, 127 F. Supp. 63, 66 (S.D.N.Y. 1954) (separate idea from expression, but only when substantiality of similarity in question); *cf.* Aliotti v. R. Dakin & Co., 831 F.2d 898, 901 (9th Cir. 1987) (the whole works should be compared, but substantial similarity should not be found when all similarities result from the use of unprotected ideas).

194. Atari, Inc. v. North Am. Philips Consumer Elecs. Corp., 672 F.2d 607, 614-15 (7th Cir.) (allowing dissection), *cert. denied,* 459 U.S. 880 (1982); Davis v. United Artists, Inc., 547 F. Supp. 722, 724 & n.8 (S.D.N.Y. 1982) (allowing dissection but no expert testimony); *see* Atari Inc. v. Amusement World, Inc., 547 F. Supp. 222, 224-25, 229-30 (D. Md. 1981) (trial court engages in dissection); Ideal Toy Corp. v. Kenner Prods., 443 F. Supp. 291, 303 (S.D.N.Y. 1977) (permitting "a certain summary of the similarities and differences which exist").

195. The Ninth Circuit has suggested that under *Kroffi,* the order of these steps should be reversed. It would first determine what aspects of the work are similar and then determine what is protected. It would find no substantial similarity when all of the similarities "arise from the use of common ideas." *Aliotti,* 831 F.2d at 901.

these approaches without relying to some degree upon expert testimony. A number of courts have admitted testimony on whether the defendant's copying constituted unlawful appropriation in the computer context.[196] With such an accommodation, this Note's approaches may be integrated readily into a conventional substantial similarity analysis.

Substantial similarity is a question of fact, and it is difficult to give abstract guidance on how similar a defendant's work must be to the expression contained in the plaintiff's work before the works may be considered *substantially* similar. A few observations, however, might be helpful. If there is a significant amount of expression in the structure of a particular computer program, one probably need not copy all of that expression to infringe the copyright in the program. In one case, nonliteral similarity with respect to only about twenty percent of a motion picture was held to be sufficient to support a finding of infringement.[197] Similarity of a relatively small proportion of the literal instructions in a program can likewise support a finding of infringement.[198] When similarity of structure coincides with literal similarity, works with somewhat less of either type of similarity may still be substantially similar.[199] When a plaintiff's work contains little protected material, only a work very similar to it, perhaps nearly a verbatim copy, may be found substantially similar.[200] The cases discussed in

196. Plains Cotton Co-op. Assn. v. Goodpasture Computer Serv., Inc., 807 F.2d 1256, 1259 (5th Cir.), *cert. denied,* 484 U.S. 821 (1987); Whelan Assocs. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1232 (3d. Cir. 1986), *cert. denied,* 479 U.S. 1031 (1987); Q-Co Indus., Inc. v. Hoffman, 625 F. Supp. 608, 610, 613 (S.D.N.Y. 1985); E.F. Johnson Co. v. Uniden Corp. of Am., 623 F. Supp. 1485, 1493 (D. Minn. 1985); *cf.* Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1136-37 (N.D. Cal. 1986) (following *Krofft* in case involving computer screen display).

Consideration of this testimony should not be undertaken lightly. After delving deeply into the mysteries of microcode, a computer language used to instruct a computer how to perform such elementary functions as adding two numbers, Judge William Gray observed:

> As I have pondered upon the testimony of the experts and studied the exhibits, I have developed a sympathetic understanding of what Judge Learned Hand meant when he observed in a relevant situation that "the more the court is led into the intricacies of dramatic craftsmanship, the less likely it is to stand upon the firmer, if more naive, ground of its considered impressions upon its own perusal."

NEC Corp. v. Intel Corp., 10 U.S.P.Q. 2d (BNA) 1177, 1184 (N.D. Cal. Feb. 6, 1989) (quoting Nicholas v. Universal Pictures Corp., 45 F.2d 119, 123 (2d Cir. 1930)). Judge Gray went to great lengths to familiarize himself with the relevant technology before hearing this case. Wirbel, *The Education of Judge Gray,* ELECTRONIC ENGINEERING TIMES, Mar. 13, 1989, at 49.

197. Universal Pictures Co. v. Harold Lloyd Corp., 162 F.2d 354 (9th Cir. 1947).

198. *In re* Certain Personal Computers and Components Thereof, 1983-1984 Copyright L. Dec. (CCH) ¶ 25,651 (Intl. Trade Commn. 1984) (similarity to 18 percent of lines in original program constitutes substantial similarity).

199. SAS Inst., Inc. v. S & H Computer Sys., 605 F. Supp. 816 (M.D. Tenn. 1985) involved 44 specific instances of literal copying, which seem to have constituted a fairly small fraction of the total lines in the program, 605 F. Supp. at 822, along with fairly comprehensive structural similarity. 605 F. Supp. at 825-26.

200. Worth v. Selchow & Righter Co., 827 F.2d 569, 573 (9th Cir. 1987), *cert. denied,* 108 S.Ct. 1271 (1988); *see* Frybarger v. Intl. Bus. Machs. Corp., 812 F.2d 525, 530 (9th Cir. 1987) (necessary expression of an idea protected only against verbatim copying).

section II.A provide some additional illustration of the kinds of facts that have supported findings of copyright infringement.

Unfortunately, the cases discussed in section II.A recite the facts in insufficient detail to decisively determine whether the results would change if this Note's techniques were applied. However, even if courts have reached correct results in past structural infringement cases, they have also announced broad rules that are more or less insensitive to the fundamental goals of copyright law. Because computer programs are utilitarian works that are also literary works, an infringement analysis involving computer programs in almost every case will implicate both the idea-expression and process-expression dichotomies. The approaches developed in this Part are rooted in these two doctrines, and when applied in tandem, isolate the part of a program's structure that is protected expression from that which is unprotected idea or process. Only by conscientiously refusing to extend copyright protection to ideas and processes can courts ensure progress in the computer art.

## CONCLUSION

Just less than ten years ago, Congress amended the copyright statute explicitly to recognize computer programs as protected works. Since then, courts have held both literal and nonliteral aspects of computer programs protected in many forms. Computer programs, as literary works that are primarily utilitarian, present special problems for copyright law. Because of their utilitarian nature, they are protected with greater doctrinal purity by patent law. Congress, however, has provided copyright protection for computer programs, and courts have been asked to determine whether this protection extends to the structure of programs. After examining the legislative history of the Copyright Act and the cases decided under the Act, this Note concludes that structure must be protected, subject to the limitations imposed by the idea-expression and process-expression dichotomies and the merger doctrine.

Courts facing the structure question have by and large recognized the problem of idea-expression merger in the context of program structure. However, while citing *Baker,* they have tended to treat the protection of structure as an all-or-nothing proposition and to give little attention to preventing the monopolization of the processes described by computer programs. They have also tended to focus their inquiry at only the highest levels of a program's structure, which would normally be expected to exhibit the least originality and be most closely tied to the function of the program. This Note proposed a pair of approaches that distinguish ideas and processes from protected expression and that probe the whole structure of a program.

This Note also illustrated the use of these approaches in a typical case of alleged structural infringement.

The approaches developed in this Note promote progress in the computer art by allowing public use of ideas and processes embodied in the structure of computer programs. As the computer art advances, courts will be asked to confront increasingly complex questions of copyright law as it relates to computer programs. It is impossible to guess at the full range of these questions. To continue promoting the progress of the computer art in the years ahead, courts must strike a proper balance between granting incentives to create new programs and hindering progress by granting excessive protection.

*— Steven R. Englund*